TUGAS AKHIR – TE 145561

MONITORING DAN SETTING DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVER CURRENT

Abi Nubli
NRP 2214038009

Dosen Pembimbing
Dr. Eng. Ardyono Priyadi, S.T., M.Eng.

PROGRAM STUDI TEKNIK LISTRIK
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya
2017

-----Halaman ini sengaja dikosongkan-----

FINAL PROJECT – TE 145561

MONITORING AND SETTING DIGITAL RELAY ONE PHASE ON OVER CURRENT

Abi Nubli
NRP 2214038009

Advisor
Dr. Eng. Ardyono Priyadi, S.T., M.Eng.

ELECTRICAL ENGINEERING STUDY PROGRAM
Electrical and Automation Engineering Department
Vocational Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2017

-----Halaman ini sengaja dikosongkan-----

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Monitoring Dan Setting Digital Relay Satu Fasa Terhadap Gangguan Over Current**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, **19 Juli 2017**

Mahasiswa

Abi Nubli
NRP 2214038009

-----Halaman ini sengaja dikosongkan-----

MONITORING DAN SETTING DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVER CURRENT

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Ahli Madya
Pada
Program Studi Teknik Listrik
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing

Dr. Eng. Ardyono Priyadi, S.T., M.Eng.
NIP. 1973 09 27 1998 03 1004

SURABAYA
JULI, 2017

-----Halaman ini sengaja dikosongkan-----

MONITORING DAN SETTING DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVER CURRENT

Nama : Abi Nubli
Pembimbing : Dr. Eng. Ardyono Priyadi, S.T., M.Eng.

ABSTRAK

Sistem kelistrikan di Indonesia saat ini di atur oleh perusahaan BUMN yang bernama PT. PLN Persero. Jadi untuk segala hal yang berkaitan dengan kelistrikan termasuk proteksi pada jaringan listrik di Indonesia yang mengurusi adalah pegawai – pegawai dari PLN. Pada proteksi jaringan listrik ini menggunakan relai dan relai ini akan bekerja bila arus yang melewati sensor relai besarnya melebihi arus yang di atur pada relai, sehingga kontak relai menutup dan mengirimkan sinyal pada *coil* PMT untuk memerintahkan PMT bekerja.

Relai ini biasa di pasang pada setiap jaringan listrik yang perlu di proteksi dan dibutuhkan koordinasi yang tepat agar relai tidak *trip* secara bersamaan. Untuk memberikan *setting* pada relai dan memantau kondisinya, operator juga harus menuju ke tempat relai itu berada agar bisa diberikan *setting* yang diperlukan.

Pada Tugas Akhir ini kami telah membuat dua buah *digital* relai untuk simulasi *trip* pada jaringan listrik satu fasa beserta koordinasinya agar tidak *trip* secara bersamaan pada saat terjadi gangguan pada salah satu saluran listrik. Relai ini dibuat *digital* agar mudah untuk melihat kondisinya secara langsung, karena terdapat LCD pada relainya yang akan menampilkan beberapa kondisi relai tersebut.

Dan agar operator tidak repot memberikan *setting* pada relai dan memantau kondisi relainya dengan datang langsung ke tempat relai itu berada, maka kami juga membuat sebuah program untuk memberikan *setting* pada kedua relai ini oleh satu komputer saja dengan jarak yang cukup jauh. Jarak ini bisa menjangkau kurang lebih 1200 meter, karena kami menggunakan media pengiriman RS485. Program ini dibuat menggunakan *software* Lazarus dengan komunikasinya dengan kedua relai menggunakan protokol Modbus.

Kata Kunci : proteksi jaringan listrik, digital relai, koordinasi relai, pengaturan jarak jauh relai, pemantauan relai

-----Halaman ini sengaja dikosongkan-----

MONITORING AND SETTING DIGITAL RELAY ONE PHASE ON OVER CURRENT

Name : **Abi Nubli**
Advisor : **Dr. Eng. Ardyono Priyadi, S.T., M.Eng.**

ABSTRACT

Electrical system in Indonesia is currently set by a company called PT. PLN Persero. So for all things related to electricity, including protection on electric power grid in Indonesia who take care is the employees of PLN. On this electric power protection using relays and relays will work when the current through the relay sensor is larger than the current set on the relay, so the relay contact closes and sends a signal to the PMT coil to order the PMT to work.

These relays are commonly installed on any power grid that needs to be protected and proper coordination is needed so that the relays do not trip simultaneously. To provide settings to the relays and monitor conditions, the operator must also go to where the relay was located in order to be given the necessary settings.

In this Final Project we have made two digital relays for trip simulation on single-phase electric power line along with its coordination so as not to trip simultaneously in the event of interference on one power line. This relay is made digital so it is easy to see its condition directly, because there is LCD in relainya which will show some condition of the relay.

And so that the operator does not bother to provide settings on relays and monitor the condition of the relay by coming directly to where the relay is located, then we also create a program to provide settings on both relays by a single computer with a considerable distance. This distance can reach approximately 1200 meters, because we use RS485 delivery media. This program was created using Lazarus software with its communication with both relays using Modbus protocol.

Keywords : *Electric power protection, digital relays, relay coordination, remote relay setting, relay monitoring*

-----Halaman ini sengaja dikosongkan-----

KATA PENGANTAR

Alhamdulillah kami panjatkan kepada Allah Subhanahu Wa Ta'ala, atas limpahan rahmat dan kemudahan dariNya, hingga kami dapat menyelesaikan tugas akhir ini dengan baik, begitu pula dengan pembuatan buku tugas akhir ini.

Tugas akhir ini dilakukan untuk memenuhi beban satuan kredit semester (SKS) yang harus ditempuh sebagai persyaratan akademis di Jurusan D3 Teknik Elektro Institut Teknologi Sepuluh Nopember Surabaya untuk menyelesaikan program pendidikan Diploma di Teknik Elektro dengan judul :

MONITORING DAN SETTING DIGITAL RELAY SATU FASA TERHADAP GANGGUAN OVER CURRENT

Penulis mengucapkan terima kasih kepada Ibu dan Bapak penulis yang memberikan berbagai bentuk doa serta dukungan tulus tiada henti, Bapak Dr. Eng. Ardyono Priyadi, S.T., M.Eng. atas segala bimbingan ilmu, moral, dan spiritual dari awal hingga terselesaikannya Tugas Akhir ini. Penulis juga mengucapkan banyak terima kasih kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung dalam proses penyelesaian Tugas Akhir ini.

Penulis menyadari dan memohon maaf atas segala kekurangan pada Tugas Akhir ini. Akhir kata, semoga Tugas Akhir ini dapat bermanfaat dalam pengembangan keilmuan di kemudian hari.

Surabaya, **19 Juli 2017**

Penulis

-----Halaman ini sengaja dikosongkan-----

DAFTAR ISI

DAFTAR GAMBAR

HALAMAN

DAFTAR TABEL

HALAMAN

-----Halaman ini sengaja dikosongkan-----

BAB I
PENDAHULUAN

1.1 Latar Belakang

Relai pada sistem kelistrikan khususnya relai arus lebih (*Over Current Relay*) mendapatkan peran yang sangat penting pada sistem proteksi saluran tenaga listrik. Relai ini akan bekerja bila arus yang melewati sensor relai besarnya melebihi arus yang diatur pada relai, sehingga kontak relai menutup dan mengirimkan sinyal pada *coil* PMT untuk memerintahkan PMT bekerja. Relai ini termasuk dalam sistem proteksi pada kelistrikan yang membutuhkan keandalan tinggi untuk menjaga keamanan pada suatu sistem. Salah satu untuk menjaga keandalannya yaitu dengan mengatur koordinasi waktu antar relai ini pada saluran listrik, agar tidak terjadi *trip* secara bersamaan pada saat ada gangguan pada salah satu saluran saja. Selain itu, relai juga merupakan komponen yang bekerja berdasarkan perubahan besarnya arus.

Semua aspek tentang sistem kelistrikan yang ada di Indonesia saat ini yang mengurusi adalah sebuah BUMN yang bernama Perusahaan Listrik Negara (PLN) dengan nama resmi PT. PLN Persero. Jadi termasuk permasalahan proteksi pada sistem kelistrikan yang mengurusi adalah PLN. Pada saat kami melakukan Kerja Praktek yang termasuk dalam mata kuliah yang ada pada semester 6 ini, kami ditempatkan di kantor PLN Area Surabaya Selatan. Dalam kerja praktek ini kami sempat mengunjungi ke sebuah Gardu Induk yang bertempat di daerah Sukolilo Surabaya. Kami disini ditunjukkan oleh pegawai PLN yang ada di sana bagaimana cara *setting* relai yang ada disana. Untuk pengaturan relai yang ada disana masih di *setting* satu persatu pada setiap relainya, dan untuk monitoring arus hanya ada tampilan pada relainya saja.

Oleh karena itu, pada Tugas Akhir ini bertujuan untuk membuat sebuah rancang bangun antar 2 relai yang sudah terkoordinasi satu sama lain dan bisa di *monitoring* dan di *setting* oleh satu komputer saja (atau bisa disebut dengan sistem *SCADA*) pada jarak yang lumayan jauh (sekitar 1200 m) tanpa harus mengatur satu persatu tiap relai. Sebenarnya untuk sistem *SCADA* pada PLN sudah diterapkan untuk pengaturan buka tutup PMT pada Gardu Induk, akan tetapi PLN belum menerapkan sistem *SCADA* pada relainya.

Untuk perancangan relai ini menggunakan mikrokontroller dengan Atmega2560 sebagai wadah program yang berfungsi sebagai *sensing* arus dan koordinasi waktu antar relai. Untuk *setting* dan *monitoring* relai ini

menggunakan protokol komunikasi Modbus. Modbus ini sudah menjadi standar protokol yang umum digunakan untuk menghubungkan peralatan elektronik industri. Beberapa alasan mengapa protokol ini banyak digunakan adalah, modbus dipublikasikan secara terbuka dan bebas *royalty*, mudah digunakan dan dipelihara, dan dapat memindahkan data *bit* atau *word* tanpa terlalu banyak membatasi *vendor*. Modbus juga mampu menghubungkan 247 peralatan (slave) dalam satu jaringan atau master tergantung dari media komunikasi yang digunakan, misalnya sebuah sistem yang melakukan pengukuran suhu dan kelembapan dan mengirimkan hasilnya ke sebuah komputer. Modbus sering digunakan untuk menghubungkan komputer pemantau dengan *Remote Terminal Unit* (RTU) pada sistem *Supervisory Control And Data Acquisition* (SCADA). Maka dari itu digunakanlan protokol Modbus.

1.2 Permasalahan

Adapun permasalahan yang akan kami angkat sebagai bahan Tugas Akhir ini :

Setting relai yang diterapkan saat ini belum ada yang menggunakan sistem *SCADA* dan masih di *setting* satu persatu tiap relainya. Maka dari itu, pada tugas akhir ini akan membuat permodelan *setting* dan *monitoring* dengan sistem *SCADA* pada 2 relai yang sudah terkoordinasi satu sama lainnya.

1.3 Batasan Masalah

Agar penulisan buku Tugas Akhir ini tidak menyimpang dan mengambang dari tujuan yang semula direncanakan sehingga mempermudah mendapatkan data dan informasi yang diperlukan, maka penulis menetapkan batasan-batasan masalah sebagai berikut :

a. *Setting* dan *monitoring* relai pada satu komputer
b. Gangguan berupa hubung singkat dan arus beban lebih
c. Simulasi *digital* relai hanya pada saluran listrik satu fasa

1.4 Tujuan

Pembuatan Permodelan *Setting Over Current Relay* yang bertujuan untuk :

Penelitian ini bertujuan untuk permodelan perancangan sistem koordinasi pada *Over Current Relay*, yang dilengkapi pemantuan dan pengaturan pada *Personal Computer* yang diharapkan mampu

mempermudah para *user* dalam mengetahui kondisi tiap relai dan juga mengatur tanpa harus menuju ke tempat relai itu berada.

1.5 Metodologi Penelitian

Tugas akhir ini yang berjudul "Monitoring Dan Setting Digital Relay Satu Fasa Terhadap Gangguan Over Current" memiliki beberapa tahap kegiatan dalam pembuatannya yaitu terdiri dari tahap persiapan (studi literatur), tahap perencanaan alat, tahap pengujian dan analisa alat, serta penyusunan laporan.

Pada tahap perencanaan (studi literatur) ini mempelajari tentang konsep pembacaan sensor arus dan tegangan pada sumber listrik AC, sistematika proteksi pada tegangan listrik satu fasa, dan mempelajari mengenai konsep kerja dan koordinasi OCR(*Over Current Relay)*.

Untuk tahap perencanaan terdiri dari *hardware* dan *software* yang meliputi perancangan *prototype* relai dilengkapi dengan sensor arus, sensor tegangan, relai DC, komunikasi RS485 dan kontaktor yang nantinya akan dikendalikan degan mikrokontroler (Arduino Mega 2560). Pada tahap ini akan dilakukan pembuatan program pada Arduino untuk pembacaan sensor arus dan tegangan yang nantinya hasil dari pembacaan sensor tersebut akan mengendalikan relai DC untuk memicu *coil* kontaktor untuk membuka dan menutup.

Setelah itu dilakukan pengujian alat, menganalisa kesalahan atau kegagalan pada alat dan mengatasi permasalahan tersebut. Tahapan ini dilakukan dengan melakukan pengujian kerja setiap relai dan koordinasi antar relai. Data hasil pengujian tersebut akan dianalisa kemudian mencari tahu faktor apa saja yang menyebabkan alat tidak bekerja sesuai dengan keinginan atau terjadi error. Tahap akhir penelitian adalah penyusunan laporan penelitian yang menjelaskan semua tahap – tahap sebelumnya.

1.6 Sistematika Laporan

Sistematika pembahasan Tugas Akhir ini terdiri dari lima bab, yaitu pendahuluan, teori penunjang, perencanaan dan pembuatan alat, pengujian dan analisa alat, serta penutup.

BAB I PENDAHULUAN

Membahas tentang latar belakang, permasalahan,batasan masalah, maksud dan tujuan, sistematika laporan, serta relevansi.

BAB II TEORI PENUNJANG

Berisi teori penunjang yang mendukung dalam perencanaan dan pembuatan alat.

BAB III PERANCANGAN ALAT

Membahas tentang perencanaan dan pembuatan perangkat keras yang meliputi rangkaian-rangkaian, desain bangun, dan perangkat lunak yang meliputi program yang akan digunakan untuk mengaktifkan alat tersebut.

BAB IV PENGUJIAN DAN ANALISA ALAT

Membahas tentang pengukuran, pengujian, dan penganalisaan terhadap kepresisian sensor dan alat yang telah dibuat.

BAB V PENUTUP

Menjelaskan tentang kesimpulan dari Tugas Akhir ini dan saran-saran untuk pengembangan alat ini lebih lanjut.

1.7 Relevansi

Diharapkan alat ini dapat terealisasi, alat ini dapat digunakan untuk Untuk mempermudah proses pembelajaran dalam koordinasi *setting Over Current Relay*.

BAB II
TEORI DASAR

Pada Bab 2 ini menjelaskan beberapa teori dasar yang menunjang dan berkaitan dengan pengerjaan Tugas Akhir ini yang berjudul "Monitoring Dan Setting Digital Relay Satu Fasa Terhadap Gangguan Over Current". Dengan adanya bahasan teori dasar ini diharapkan bisa membantu menyelesaikan Tugas Akhir ini dan juga dapat dijadikan referensi untuk beberapa bahan yang digunakan dalam Tugas Akhir ini.

2.1 Arduino MEGA 2560

Arduino adalah pengendali mikro *single-board* yang bersifat *open-source*, dan dirancang untuk memudahkan penggunaan elektronik dalam berbagai bidang. *Hardware*-nya memiliki prosesor Atmel AVR dan *software*-nya memiliki bahasa pemrograman sendiri. Arduino ini sendiri memiliki banyak jenis, salah satunya yang kami gunakan dalam pengerjaan tugas akhir ini yaitu Arduino mega 2560.

Arduino Mega 2560 adalah papan mikrokontroler berdasarkan ATmega2560. Arduino ini memiliki 54 digital pin input / output (yang 15 dapat digunakan sebagai output PWM), 16 analog input, 4 UART (hardware port serial), 16 MHz osilator kristal, koneksi USB, jack listrik, header ICSP, dan tombol reset. Arduino ini berisi semua yang diperlukan untuk mendukung mikrokontroler; untuk dapat terhubung ke komputer dengan meggunakan kabel USB atau sumber tegangan berasal dari adaptor AC-DC atau baterai untuk menghidupkan arduino .

Gambar 2.1 *Board* Arduino MEGA 2560

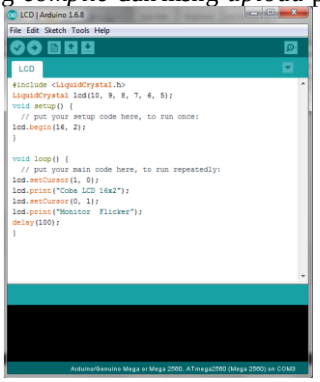
Spesifikasi Arduino Mega 2560 adalah sebagai berikut:

1. Menggunakan chip microcontroller AtMega2560.

2. Tegangan operasi 5 Volt.
3. Tegangan input (yang direkomendasikan, via jack DC) sebesar 7-12 Volt. Untuk batas maksimal input sebesar 6-20 Volt.
4. Digital I/O sebanyak 54 buah, 6 diantaranya menyediakan PWM output.
5. Analog input pin sebanyak 16 buah.
6. Arus DC per pin I/O sebesar 20 mA.
7. Arus DC pada pin 3,3 Volt sebesar 50 mA.
8. Flash memory sebesar 256 KB, 8 KB telah digunakan untuk bootloader.
9. SRAM sebesar 8 kb.
10. EEPROM sebesar 4 kb.
11. Clock speed sebesar 16 Mhz.
12. Dimensi Arduino Mega 2560 sebesar 101,5 mm x 53,4 mm.
13. Berat Arduino Mega 2560 sebesar 37 g.

2.2 *Software* Arduino IDE

Board Arduino dapat diprogram menggunakan *software open source* bawaan Arduino IDE. Arduino IDE adalah sebuah aplikasi *crossplatform* yang berbasis bahasa pemrograman *processing* dan *wiring*. Arduino IDE didesain untuk mempermudah pemrograman dengan adanya kode editor yang dilengkapi dengan *syntax highlighting*, *brace matching*, dan indentasi otomatis untuk kemudahan pembacaan program, serta dapat meng-*compile* dan meng-*upload* program ke *board* dalam satu klik.



Gambar 2.2 Jendela Arduino IDE

IDE Arduino adalah *software* yang sangat canggih ditulis dengan menggunakan Java. IDE Arduino terdiri dari:

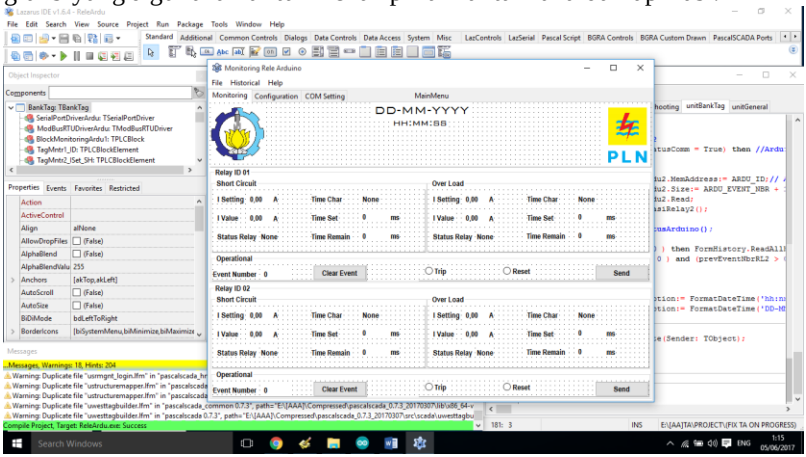
1. *Editor* program, sebuah *window* yang memungkinkan pengguna menulis dan mengedit program dalam bahasa *processing*.
2. *Compiler*, sebuah modul yang mengubah kode program (bahasa *Processing*) menjadi kode biner. Bagaimanapun sebuah *microcontroller* tidak akan bisa memahami bahasa *processing*. Yang bisa dipahami oleh *microcontroller* adalah kode biner. Itulah sebabnya *compiler* diperlukan dalam hal ini.
3. Uploader, sebuah modul yang memuat kode biner dari komputer ke dalam *memory* di dalam papan Arduino.

2.3 *Software* Lazarus

Lazarus adalah Integrated Development Environment (IDE) yang bersifat *open source* bagi pengguna bahasa pemrograman Pascal dan yang menyediakan lingkungan pengembangan yang mirip dengan Delphi yaitu untuk mengembangkan aplikasi konsol, desktop, web, ataupun perangkat mobile. LPT ini dibangun untuk dan didukung oleh kompilator Free Pascal (FPC). Mempunyai moto Write Once Compile Anywhere artinya hanya dengan sebuah kode sumber program dapat dikompilasi di semua platform OS (Windows, Linux, Mac OS dan lain-lain.) dan arsitektur (i386, x86_64, arm dan lain-lain) yang didukung.

Gambar 2.3 Tampilan Awal Lazarus

Lazarus dibangun di atas kerangka yang juga digunakan untuk menghasilkan aplikasi yang dibuat di atasnya, yaitu Lazarus Component Library (LCL). LCL ini merupakan abstraksi untuk berbagai pustaka grafis yang digunakan untuk menampilkan antarmuka dari aplikasi.



Gambar 2.4 Tampilan *Software* IDE Lazarus

Dari gambar 2.4 di atas ini menunjukkan *Software* IDE Lazarus yang digunakan sebagai *interface* pada Tugas Akhir kami. Pada IDE Lazarus ini terdiri dari *menubar, toolbar, component palette, object inspector, source editor, form designer dan compiler messages.* Berikut adalah penjelasannya:

1. Bagian Atas Lazarus

   Bagian atas pada Lazarus ini berisi komponen – komponen yang berfungsi untuk membantu pembuatan pada IDE Lazarus. Bagian ini terdiri dari *menubar, toolbar, dan component messages.*

Gambar 2.5 Tampilan Bagian Atas Lazarus

   a. *Menubar*

   Menubar berisi perintah - perintah yang digunakan untuk membuka, menyimpan, mengubah *option*. Sebagian dari fungsi *menubar* juga dapat dipanggil dengan tombol pada *toolbar*, maupun dengan cara menekan/ klik tombol - tombol fungsi

pada *keyboard*, diantaranya yang paling sering digunakan adalah tombol fungsi "F9" yang berfungsi untuk menjalankan aplikasi/ program yang sudah di buat. Tombol fungsi "F11" berfungsi untuk menampilkan *object inspector*, "F12" berfungsi untuk menampilkan secara bergantian (toggle view) *form*/ *unit*. Untuk menampilkan jendela *form* maupun *source editor* maka tekan "Shift+F12" untuk menampilkan Form, dan "Ctrl+F12" untuk menampilkan Source Editor. Bagian *menubar* terletak pada bagian paling atas pada gambar 2.5 di atas.

b. *Toolbar*

Toolbar juga dapat melakukan beberapa operasi seperti pada *menubar* hanya dengan mengklik sebuah tombol. Setiap tombol pada *toolbar* mempunyai sebuah *hint* yang berisi informasi mengenai fungsi dari tombol tersebut. Bagian *toolbar* terletak pada bagian bawah paling kiri pada gambar 2.5 di atas.

c. *Component Palette*

Component Palette adalah *toolbar* yang berisi komponen-komponen yang terkumpul dalam *tab* - *tab* dalam kategori yang sama dan berfungsi untuk membangun program aplikasi. Isi dari *component palette* ini dapat ditambah sesuai kebutuhan. Di *component palette* inilah pemrograman *visual* Lazarus *Free Pascal* kelihatan nyata sebagai pemrograman *visual* yang memudahkan penggunanya, tinggal klik komponennya dan meletakkan dalam *form* saja.

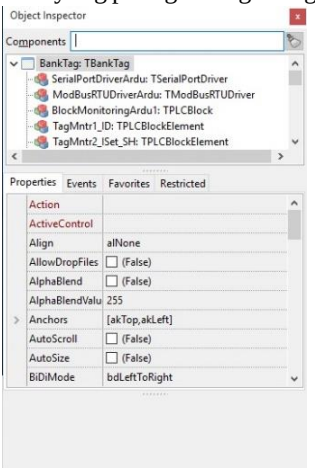
2. *Object Inspector*

Dengan *object inspector*, kita dapat merubah properti dari setiap komponen dengan mudah sekali. Jika kita dapat mengontrol tindakan yang diambil jika terjadi *event*. *Object inspector* terdapat 4 *tab*, yaitu *properties*, *events*, dan *favorites*.

Tab properties memberikan fasilitas untuk melihat dan mengubah properti dari setiap item. Isi dari *object inspector* berubah - ubah mengikuti komponen yang dipilih. Jika kita melihat tanda + (plus), berarti properti tesebut mempunyai subproperti.

Tab events berisi *event* - *event* yang dapat direspon oleh sebuah objek. Klik *tab events* di sebelah kanan *tab properties*. Misalnya saat

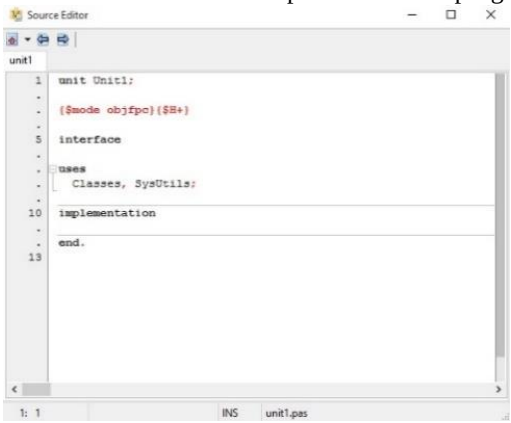
ada sesuatu yang dikerjakan pada saat form aktif, maka pada *event* harus dinyatakan *OnActivate*. Untuk *tab favorites* hanya berisi event-event yang paling sering kita gunakan.



Gambar 2.6 Tampilan *Object Inspector*

3. *Source Editor*

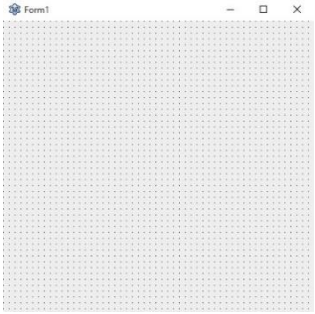
Source editor merupakan bagian terpenting di Lazarus. Jendela ini dipakai untuk menuliskan program *Free Pascal*. Editor Lazarus sangat canggih, dengan fasilitas - fasilitas *highlight*, *autocomplete* untuk memudahkan dalam penulisan kode program.



Gambar 2.7 Tampilan *Source Editor*

4. *Form Designer*

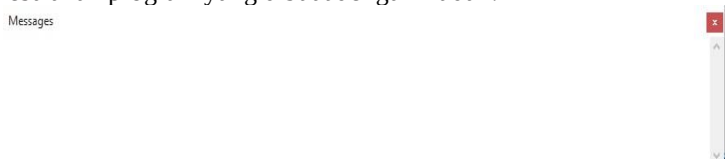
Form designer ini diawali dengan jendela kosong dengan nama *default* Form1 yang digunakan untuk merancang aplikasi secara *visual*. Dari sini kita sudah bisa menentukan tampilan aplikasi yang akan dibuat. Kita juga berinteraksi pada *form designer* dengan cara memilih komponen dari *component palette* dan meletakkannya ke dalam *form*. Setelah ada komponen dalam *form*, kita dapat mengatur posisi atau mengubah ukurannya. Kita dapat merubah tampilan dan perilaku komponen dengan menggunakan *object inspector* dan *source editor*.



Gambar 2.8 Tampilan *Form Designer*

5. *Compiler Messages*

Compiler messages berfungsi untuk menyatakan informasi yang akan diberikan oleh *compiler* Lazarus. Di dalam *compiler messages* nanti akan ditampilkan *hints* dan atau peringatan selama proses *compile*. Dengan membaca pesan disini kita bisa memperbaiki kesalahan program yang dibuat dengan mudah.



Gambar 2.9 Tampilan *Compiler Messages*

2.4 *Relay* DC 5 V AZ942

Relay adalah Saklar (*Switch*) yang dioperasikan secara listrik dan merupakan komponen elektromaknetik yang terdiri dari 2 bagian utama yakni elektromagnet (c*oil*) dan mekanikal (seperangkat kontak

saklar/s*witch*). Relay menggunakan prinsip elektromagnetik untuk menggerakkan kontak saklar sehingga dengan arus listrik yang kecil (*low power*) dapat menghantarkan listrik yang bertegangan lebih tinggi.

Gambar 2.10 *Relay* AZ942

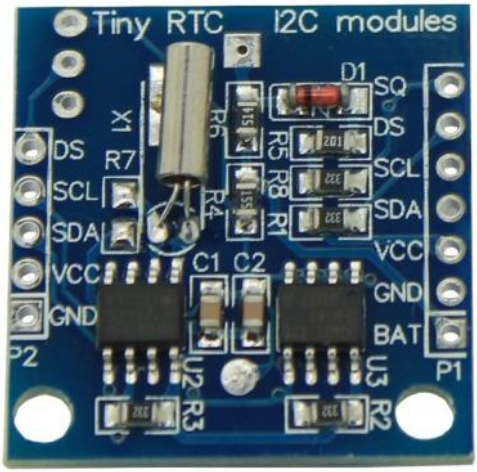
AZ942 merupakan *relay* yang bekerja berdasarkan prinsip elektromagnetik, dengan tegangan operasi pada *coil* sebesar 5VDC dan arus sebesar 71,4 mA. Dengan tegangan 5 VDC, dapat membangitkan tegangan pada kontak sebesar 30VDC dan 277VAC. waktu operasi pada relay ini adalah 10ms (maks) dan waktu rilisnya 5ms(maks).

2.5 Modul *Real Time Clock* (RTC) DS1307

Real Time Clock (RTC) adalah jenis pewaktu yang bekerja berdasarkan waktu yang sebenarnya atau dengan kata lain berdasarkan waktu yang ada pada jam kita. Meskipun istilah sering mengacu pada perangkat di komputer pribadi, server dan embedded system, RTC hadir di hampir semua perangkat elektronik yang perlu untuk menjaga keakuratan waktu. RTC memiliki sumber tenaga alternatif, sehingga mereka dapat terus menjaga waktu sementara sumber utama daya mati atau tidak tersedia. Sumber tenaga alternatif ini biasanya berupa baterai lithium dalam sistem lama, tetapi beberapa sistem yang lebih baru menggunakan supercapacitor, karena mereka dapat diisi ulang dan dapat disolder. Sumber daya alternatif juga dapat menyalurkan listrik ke RAM yang didukung baterai. Pada umumnya tenaga alternatif yang digunakan sebesar 3 Volt dari baterai lithium.

Kebanyakan RTC menggunakan osilator kristal, tetapi beberapa menggunakan frekuensi saluran listrik. Dalam banyak kasus frekuensi osilator yang digunakan adalah 32,768 kHz. Frekuensi ini sama dengan yang digunakan dalam jam kuarsa dan jam tangan, selain itu frekuensi

yang dihasilkan adalah persis 215 siklus per detik, yang merupakan tingkat nyaman untuk digunakan dengan sirkuit biner sederhana.



Gambar 2.11 RTC Tiny I2C *modules*

Modul RTC kecil ini didasarkan pada chip jam DS1307 yang mendukung protokol I2C. RTC ini menggunakan sel baterai Lithium (CR1225). Jam / kalender menyediakan detik, menit, jam, hari, tanggal, bulan, dan Informasi tahun. Akhir tanggal bulan secara otomatis disesuaikan dengan bulan dengan kurang dari 31 hari, termasuk koreksi untuk tahun kabisat. Jam beroperasi baik dalam 24 jam atau 12 jam dengan format AM / PM indikator.

2.6 Modul SD *Card Shield*

Modul SD *Card* merupakan suatu modul yang digunakan untuk mempermudah antarmuka antara SD *Card* dan mikrokontroler dengan tegangan kerja +5 VDC. SD *Card* dapat digunakan sebagai memori yang dapat diganti dengan mudah sehingga memudahkan dalam ekspansi ke kapasitas memori yang lebih besar. Tersedia Ferroelectric Nonvolatile RAM (FRAM) yang dapat digunakan sebagai *buffer* sementara dalam mengakses SD *Card* atau sebagai tempat penyimpan data lain. Modul ini dapat digunakan antara lain sebagai penyimpan data pada sistem absensi, sistem antrian, atau aplikasi *datalogging* lainnya.

Gambar 2.12 Modul SD *Card Shield*

Modul SD *Card Shield* ini memiliki spesifikasi kerja h*ardware* sebagai berikut :

1. Tegangan *supply* +5 VDC.
2. Jenis kartu yang didukung: SD *Card* (dan MMC).
3. Antarmuka SD *Card* (dan MMC) dengan mikrokontroler secara SPI.
4. Tersedia 2 KByte Ferroelectric Nonvolatile RAM FM24C16.
5. Antarmuka FRAM dengan mikrokontroler secara *TwoWire Interface*.
6. Tersedia contoh aplikasi untuk DT-51™ *Low Cost Series* dan DT-AVR *Low Cost Series* dalam bahasa *BASIC* untuk MCS-51® (BASCOM-8051©) dan bahasa C untuk AVR® (CodeVisionAVR©).
7. Kompatibel dengan DT-51™ *Low Cost Series* dan DTAVR *Low Cost Series*. Mendukung DT-51™ *Minimum System* (MinSys) ver 3.0, DT-51™ PetraFuz, dan lain-lain.

2.7 Modul *Liquid Crystal Diplay* (LCD) *Keypad Shield*

LCD (*Liquid Crystal Diplay*) berfungsi menampilkan suatu nilai hasil sensor, menampilkan teks, atau menampilkan menu pada aplikasi microcontroller. Sumber cahaya di dalam sebuah perangkat LCD adalah lampu neon berwarna putih dibagian belakang susunan kristal cair tadi. Titik cahaya yang jumlahnya puluhan ribu bahkan jutaan inilah yang membentuk tampilan. Kutub kristal cair yang dilewati arus listrik akan

berubah karena pengaruh polarisasi medan magnetik yang timbul dan oleh karenanya akan hanya beberapa warna.

LCD membutuhkan driver supaya bisa dikoneksikan dengan sistem minimum dalam suatu microcontroller. Driver yang disebutkan berisi rangkaian pengaman, pengatur tingkat kecerahan maupun data, serta untuk mempermudah pemasangan di microcontroller.

Gambar 2.13 *Liquid Crystal Diplay* (LCD) *Keypad*

LCD *Keypad* dikembangkan untuk Arduino *Shield*, tujuannya untuk menyediakan antarmuka yang *user-friendly* dan memungkinkan pengguna untuk membuat berbagai pilihan menu dan lainnya. LCD *Keypad* ini terdiri dari 1602 karakter putih *backlight* biru. Terdapat 5 tombol yang terdiri dari *select, up, right, down* dan *left*. Untuk menyimpan pin IO digital, antarmuka keypad hanya menggunakan satu saluran ADC. Berikut adalah konfigurasi pin untuk LCD *Keypad*.

Tabel 2.1 Konfigurasi Pin LCD *Keypad* 16x2

Pin	Function
Analog 0	Button (select, up, right, down and left)
Digital 4	DB4
Digital 5	DB5
Digital 6	DB6
Digital 7	DB7
Digital 8	RS (Data or Signal Display Selection)
Digital 9	Enable
Digital 10	Backlit Control

2.8 Modul Komunikasi RS485

Modul RS485 atau *Electronic Industries Association* (EIA) RS485 adalah jaringan *balanced line* dan dengan sistem pengiriman data secara

half-duplex. *Half-duplex* ini berarti bisa mengirim atau menerima pada kabel yang sama, tetapi hanya dalam satu arah dan pada satu waktu yang sama. Modul ini hanya mempunyai dua sinyal, A dan B dengan perbedaan tegangan antara keduanya. Karena line A sebagai referensi terhadap B maka sinyal akan high bila mendapat input low demikian pula sebaliknya. Pada komunikasi RS485, semua peralatan elektronik berada pada posisi penerima hingga salah satu memerlukan untuk mengirimkan data, maka peralatan tersebut akan berpindah ke mode pengirim, mengirimkan data dan kembali ke mode penerima. Setiap kali peralatan elektronik tersebut hendak mengirimkan data, maka terlebih dahulu harus diperiksa, apakah jalur yang akan digunakan sebagai media pengiriman data tersebut tidak sibuk. Apabila jalur masih sibuk, maka peralatan tersebut harus menunggu hingga jalur sepi.

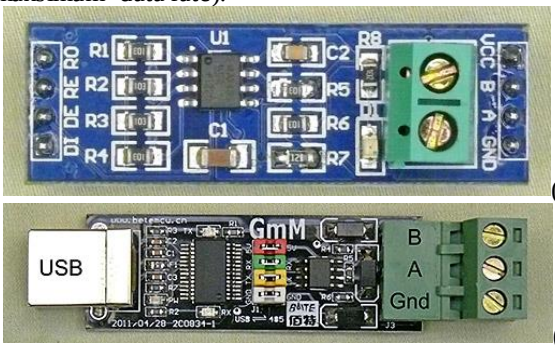
Agar data yang dikirimkan hanya sampai ke peralatan elektronik yang dituju, misalkan ke salah satu *slave*, maka terlebih dahulu pengiriman tersebut diawali dengan *slave ID* dan dilanjutkan dengan data yang dikirimkan. Peralatan elektronik yang lain akan menerima data tersebut, namun bila data yang diterima tidak mempunyai *ID* yang sama dengan *slave ID* yang dikirimkan, maka peralatan tersebut harus menolak atau mengabaikan data tersebut. Namun bila *slave ID* yang dikirimkan sesuai dengan *ID* dari peralatan elektronik yang menerima, maka data selanjutnya akan diambil untuk diproses lebih lanjut.

Tabel 2.2 Tabel Karakteristik RS485

Parameter	**RS485**
Differential	yes
Max number of drivers Max number of receivers	32 32
Modes of operation	half duplex
Network topology	multipoint
Max distance (acc. standard)	1200 m
Max speed at 12 m Max speed at 1200 m	35 Mbs 100 kbs
Max slew rate	n/a
Receiver input resistance	≧ 12 kΩ

Parameter	RS485
Driver load impedance	54 Ω
Receiver input sensitivity	±200 mV
Receiver input range	–7..12 V
Max driver output voltage	–7..12 V
Min driver output voltage (with load)	±1.5 V

Pada tabel di atas menunjukkan bahwa RS485 merupakan standar antarmuka yang unik, karena dari semua standar EIA hanya RS485 yang dapat digunakan untuk operasi multiple driver. Oleh karena itu memungkinkan konfigurasi *multipoint* (party line). Saluran komunikasi *multipoint* ini dapat dihubungkan sampai dengan 32 *driver/generator* dan 32 *receiver* pada dua kabel (two wires). RS485 ini mempunyai sifat yang sangat kebal terhadap gangguan listrik, sehingga dapat menyalurkan data lebih jauh yaitu maksimal 1,2 km (4000 feet) dengan kecepatan lebih tinggi (100 kbps) atau 1200 m (40 feet) dengan kecepatan 10Mbps (maksimum data rate).



(a)

(b)

Gambar 2.14 Modul Komunikasi RS485, (a) Modul RS485 Arduino, (b) Modul USB to RS485

Pin yang ada pada RS485 adalah sebagai berikut :

a) DI (*Data In*) Data pin DI ditransmisikan pada baris A & B saat modul berada dalam mode *Transmit* (mengirim data). Untuk mengatur modul dalam mode pengiriman nilai DE dibuat 1 dan RE dibuat 1. Pin DI terhubung ke pin Tx *Host Microcontroller* UART.

b) RE (*Receive Enable*) RE pin digunakan untuk mengkonfigurasi modul dalam *Receive Mode* (menerima data).
c) DE (*Data Enable*) DE pin digunakan untuk mengkonfigurasi modul dalam *Mode Transmitt*. Untuk mengfungsikan RS485 dalam mode *Transmit* dan *Receive* pin DE dan RE dihubungkan menjadi 1.
d) RO (*Receive Out*) data yang diterima pada pin A & B diberikan pada pin RO. Pin RO terhubung ke pin Rx dari mikrokontroler.
e) A & B (*Differential Input and Ouput Pins*) data ditransmisikan dan diterima pada garis A & B.

2.9 *Setting Relay*

Aplikasi perhitungan dahulu digunakan untuk menentukan pengaturan yang sesuai pada relai konvensional, namun untuk relai saat ini sudah banyak yang memakai relai digital dimana pengaturan dapat diatur dan dikontrol dengan *microcontroller*. Tetapi perhitungan pengaturan relai OCR tetap dilakukan sebagai ilmu dasar untuk melakukan teknik men-*setting* suatu relai.

Berikut merupakan rumus waktu kerja relai yang dipakai :

$$\boldsymbol{t = \frac{\alpha \; x \; tms}{\{(\frac{If}{Iset})^{\beta} - 1\}}}$$

$$\boldsymbol{tms = \frac{t \; x \; \{(\frac{If}{Iset})^{\beta} - 1\}}{\alpha}}$$

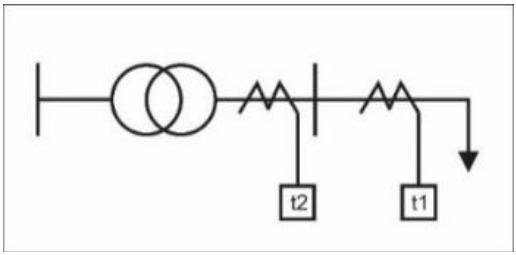
Keterangan :

If = Arus gangguan
Iset = Arus *setting*
tms = time dial
α = koefisien kali
β = koefisien pangkat

Tabel 2.3 Tabel Karakteristik *Relay*

KARAKTERISTIK RELAY	α	β
STANDARD INVERSE	0.14	0.02
LONG TIME INVERSE	120	1
EXTREMELY INVERSE	80	2
VERY INVERSE	13.5	1
SHORT TIME INVERSE	0.05	0.04

Tabel 2.3 menunjukkan nilai α dan β dari setiap karakteristik relai. Sedangkan untuk relai yang penulis hitung adalah relai pada sisi sekunder yaitu Relai Arus Lebih pada penyulang dan PMCB. Untuk gambar 2.11 diagram satu garis koordinasi relai dapat dilihat pada gambar di bawah ini.



Gambar 2.15 Sistem Koordinasi *Over Current Relay*

Sebagai catatan bahwa nilai waktu kerja yang paling cepat adalah relai yang paling jauh dari sumber, ini berarti waktu kerja t1 lebih cepat dibanding t2.

-----Halaman ini sengaja dikosongkan-----

BAB III
PERENCANAAN ALAT

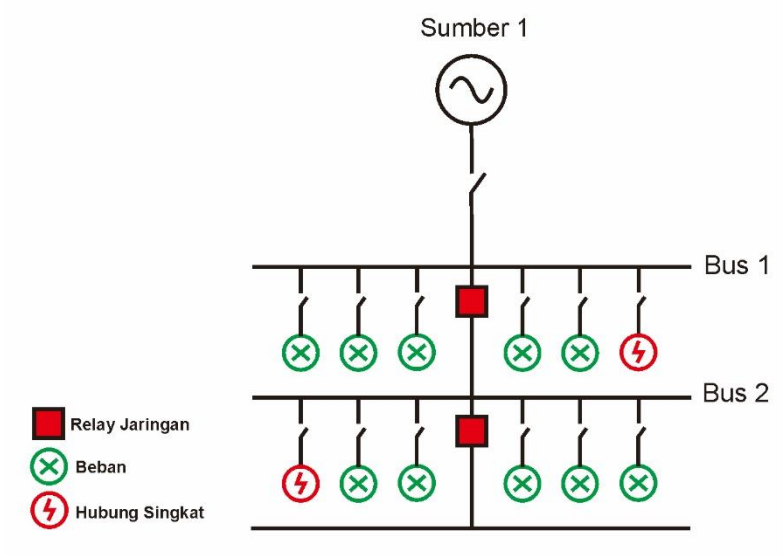
Pada bab 3 ini membahas mengenai perencanaan alat yang terdiri dari perancangan serta pembuatan alat pada Tugas Akhir kami yang berjudul **"Monitoring Dan Setting Digital Relay Satu Fasa Terhadap Gangguan Over Current".** Dalam perancangan dan pembuatan alat ini dibagi menjadi dua yaitu pada perangkat elektronik (*hadware*) dan pada perangkat lunak (*software*). Untuk bagian saya meliputi:

1. Perancangan *Hardware* terdiri dari :
   a Perancangan LCD *Keypad Shield*
   b Perancangan Modul Komunikasi RS485
   c Perancangan Modul Rangkaian RTC
   d Perancangan Modul Rangkaian SD *Card*
2. Perancangan *Software* yang berupa *flowchart* terdiri dari :
   a Pemrograman Arduino IDE
   b Pemrograman LCD *Keypad Shield*
   c Pemrograman RTC
   d Pemrograman SD *Card*
   e Pemrograman *Interface* dengan Lazarus

3.1 Diagram Fungsional Alat

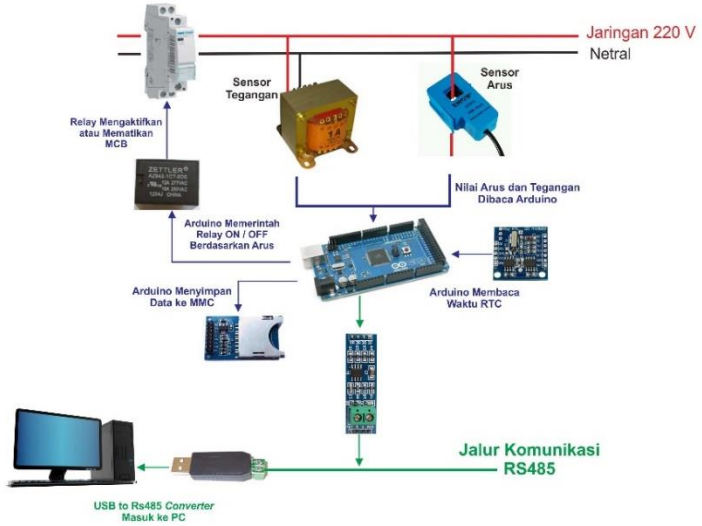
Perencanaan alat pada Tugas Akhir ini yang berjudul "Monitoring Dan Setting Digital Relay Satu Fasa Terhadap Gangguan Over Current" membahas tentang sistem kerja keseluruhan dari alat kami. Dan pada alat kami ini membahas tentang dua buah *digital relay* yang sudah dikoordinasikan satu sama lainnya dan berfungsi untuk mengamankan jaringan listrik dari adanya arus lebih yang disebabkan hubung singkat dan beban lebih.

Yang dimaksudkan terkoordinasi disini yaitu pada kedua relai sudah memiliki waktu dan arus *setting* masing – masing yang sudah diperhitungkan terlebih dahulu *trip* nya agar nantinya kedua relai ini dapat mengamankan jaringan yang memiliki gangguan saja tanpa mengganggu jaringan listrik lainnya yang masih sehat. Selain untuk mengamankan jaringan listrik, alat pada tugas akhir kami juga dapat memantau kondisi dari kedua relai dan juga memberikan *setting* pada kedua relai dari jarak yang cukup jauh dengan diatur oleh satu komputer saja.



Gambar 3.1 Skema Sistem Secara Keseluruhan

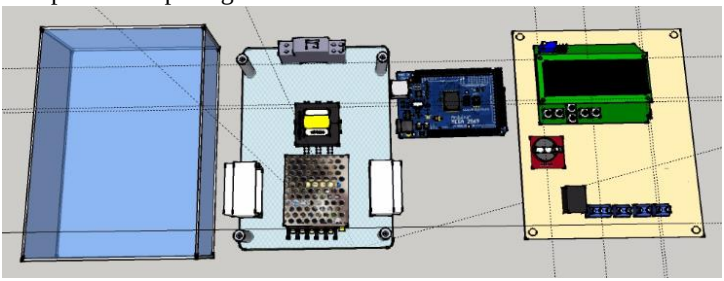
Sistem kerja dari relai ini dimulai dari memberikan *setting* pada relai yang terdiri dari *setting* arus, waktu, dan tipe karakteristik yang akan digunakan. Selanjutnya mengambil nilai dari sensor arus dan sensor tegangan, setelah terdeteksi bahwa arus pada jaringan adalah arus nominal, maka MCB akan *close* dan menyambungkan jaringan. Tetapi jika sensor membaca nilai dari arus nominal lebih kecil daripada arus *setting* maka MCB akan trip sesuai dengan waktu dan tipe karakteristik yang digunakan. Selanjutnya RTC akan membaca waktu dan tanggal secara *real time*, ini digunakan untuk pendataan kondisi dari jaringan agar dapat dipantau *user* secara *real time*. LCD pada relai akan menampilkan tegangan dan arus saat ini yang terukur dari jaringan. Kondisi tersebut akan disimpan di data *logger* dan akan dikirimkan ke PC untuk memonitor secara langsung. Pada PC ini dengan menggunakan software Lazarus akan menampilkan nilai arus nominal dan juga *setting* dari relai yang digunakan dan juga bisa memberi *setting* lewat Lazarus jika perlu.



Gambar 3.2 Skema Relai

3.2 Perancangan Elektronik

Pada perancangan elektronik (*hardware*) ini komponennya meliputi Rangkaian Sensor Tegangan, Rangkaian Sensor Arus, LCD Keypad 16 x 2, Komunikasi RS485, Relay dan MCB, Rangkaian RTC, Rangkaian SD Card, Shield Arduino Mega. Tampilan perancangan *hardware* untuk relai utama dapat dilihat pada gambar di bawah ini.



Gambar 3.3 Perancangan Elektronik Relai

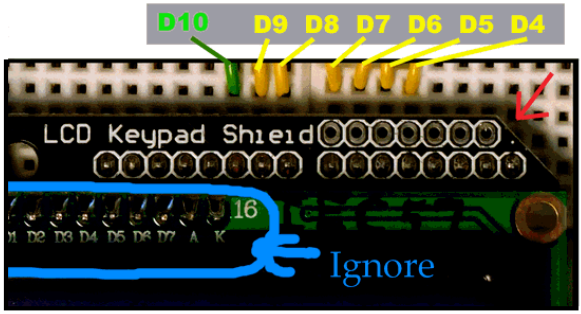
Susunan dari perancangan elektronik terbagi menjadi 2 bagian, yaitu, bagian atas dan bagian bawah. Pada bagian bawah terdapat *Power Supply*, CT Sensor, Trafo sebagai sensor tegangan, MCB dan *Stop Contact* untuk

menghubungkan jaringan dengan MCB yang ada pada relai. Sedangkan pada bagian atas terdapat 2 sisi bagian, pada sisi bawah terdapat *board* Arduino sebagai kontrol pada Tugas Akhir ini, sedangkan pada sisi atas terdapat rangkaian Sensor Tegangan, rangkaian Sensor Arus, rangkaian Relai penggerak MCB, LCD *Keypad* 16 x 2, Komunikasi RS485, RTC dan *SD Card*.

3.2.1 Perancangan LCD *Keypad Shield*

Pada tugas akhir ini kami menggunakan *Liquid Crystal Display* (LCD) *keypad* yang digunakan untuk mempermudah *user* untuk memantau kondisi terkini dari relai. LCD *keypad shield* ini dapat dipasang langsung di atas *board* arduino termasuk arduino mega 2560 yang kami gunakan, tanpa menyolder dan tanpa kabel. LCD *keypad shield* yang kami gunakan ini menggunakan *chip* HDD44780 dan merupakan LCD karakter 16 x 2. Pada LCD *keypad shield* ini akan dimasukkan program untuk menampilkan beberapa menu yang menunjukkan kondisi terkini dari relai. Menu yang akan ditampilkan ada 3, berikut adalah penjelasannya:

1. Menu *Display* yang menampilkan I *setting*, I rms dan V rms.
2. Menu Status yang menampilakan Waktu terkini dari RTC dan Status *trip* dari Relai.
3. Menu *Trip-Reset* yang berisi perintah *Test Trip*, *Reset* dan untuk kembali ke Menu Awal.



Gambar 3.4 Rangkaian LCD *Keypad Shield* dengan Arduino

Berikut adalah pengkabelan yang harus dilakukan untuk menghubungkan LCD *Keypad Shield* dengan Arduino Mega 2560 :

1. Pin A0 dengan pin 5 tombol LCD *Keypad*
2. Pin Digital 4 dihubungkan dengan pin DB 4
3. Pin Digital 5 dihubungkan dengan DB5

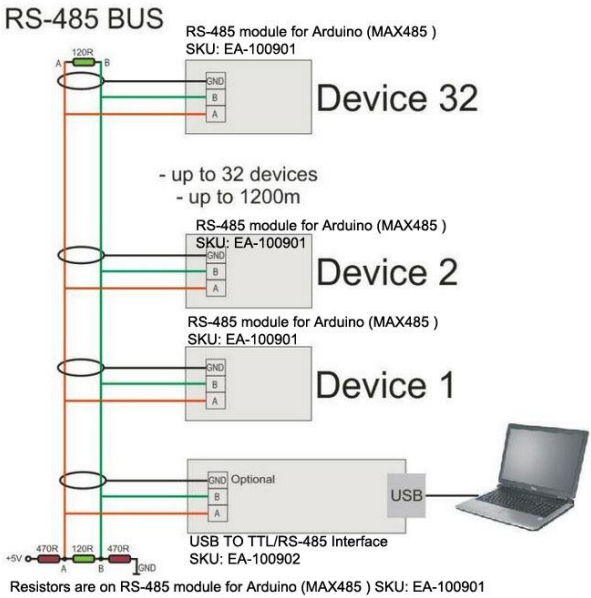
4. Pin Digital 6 dihubungkan dengan DB 6
5. Pin Digital 7 dihubungkan dengan DB7
6. Pin Digital 8 dihubungkan dengan pin RS
7. Pin Digital 9 dihubungkan dengan pin *Enable*
8. Pin Digital 10 dihubungkan dengan pin *Backlight Control*

3.2.2 Perancangan Modul Komunikasi RS485

Dalam perancangan tugas akhir ini kami memerlukan komunikasi yang digunakan untuk koordinasi antar relai saat *trip* pada salah satu saluran dan juga komunikasi antara kedua relai dengan *interface* Lazarus yang ada pada komputer. Modul komunikasi yang digunakan harus dapat mengirimkan data dengan jarak yang cukup jauh dan juga bisa digunakan untuk berkomunikasi antara master dengan beberapa slave, karena dalam kondisi di lapangan relai akan dipasang tidak berdekatan satu sama lain dan juga dengan komputer yang berperan sebagai *master*.

Maka dari itu kami menggunakan komunikasi menggunakan RS485 yang mampu mentransmisikan data hingga jarak maksimum ±1200 m atau 1,2 km. Data ditransmisikan oleh 2 buah kabel, yaitu A dan B. Untuk dapat mencapai transmisi data sepanjang 1,2 km, media pembawa data harus lancar dan panjang garis A dan B harus jarak jauh.

Untuk menyambungkan RS485 dengan arduino menggunkanan pin Rx, Tx dan pin digital. Pada Arduino Mega, pin DI (*data in*) RS485 terhubung ke pin 18 Arduino sebagai Tx1 *software serial*. Pin RO (*receive out*) ke pin 19 Arduino sebagai Rx1 *software serial*. Pin DE (*data enable*) dan RE (*receive enable*) dijumper dan dihubungkan ke pin 8 Arduino. Pin A dan B (RS485 *pair*) dipasangkan kedua modul RS485. Dan untuk Vcc dan GND RS485 dihubungkan pada Vcc dan GND Arduino. Berikut adalah gambar sambungan antara Arduino (slave) dengan komputer (master) menggunakan RS485.



Gambar 3.5 Sambungan RS485 antara komputer (Master) dengan beberapa Arduino (Slave)

Komputer sebagai *master* ini berfungsi untuk memberikan data *setting* dan juga *monitoring* data – data yang ada pada relai (*slave)*. Sistem seperti ini dinamakan dengan *Supervisory Control And Data Acquisition* (SCADA). Pengaturan sistem tenaga listrik yang komplek, sangat bergantung kepada SCADA. Pengaturan sistem tenaga listrik dapat dilakukan secara manual ataupun otomatis. Pada pengaturan secara manual, operator mengatur pembebanan pembangkit dengan melihat status peralalatan listrik yang mungkin dioperasikan misalnya *Circuit Breaker* (CB), beban suatu pembangkit, beban trafo, beban suatu transmisi atau kabel dan mengubah pembebanan sesuai dengan frekuensi sistem tenaga listrik. Pengaturan secara otomatis dilakukan dengan suatu aplikasi yang bisa mengatur pembebanan pembangkit berdasar *setting* yang dihitung terhadap simpangan frekuensi.

Salah satu hal yang penting pada sistem SCADA adalah komunikasi data antara sistem *remote* (remote station / RTU) dengan pusat kendali. Komunikasi pada sistem SCADA mempergunakan protokol khusus,

walaupun ada juga protokol umum yang dipergunakan. Protokol yang kami gunakan ini bernama Modbus, khususnya Modbus RTU. Modbus ini adalah protokol komunikasi serial yang dipublikasikan oleh Modicon pada tahun 1979 untuk diaplikasikan ke dalam *programmable logic controllers* (PLCs). Modbus sudah menjadi standar protokol yang umum digunakan untuk menghubungkan peralatan elektronik industri.

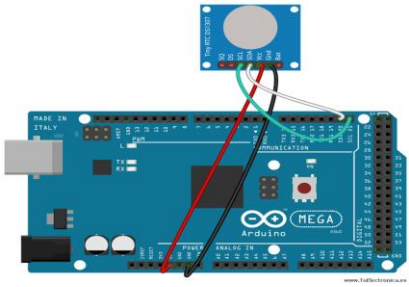
Keunggulan dari Modbus ini banyak digunakan antara lain, modbus dipublikasikan secara terbuka dan bebas royalty, mudah digunakan dan modbus mampu menghubungkan 247 peralatan (slave) dalam satu jaringan atau *master*, misalnya sebuah sistem yang melakukan pengukuran suhu dan kelembapan dan mengirimkan hasilnya ke sebuah komputer. Modbus sering digunakan untuk menghubungkan komputer pemantau dengan *remote terminal unit* (RTU) pada sistem *supervisory control and data acquisition* (SCADA). Modbus mempunyai beberapa jenis salah satu yang kami gunakan sebgai protocol yaitu, Modbus RTU yang merupakan varian modbus yang ringkas dan digunakan pada komunikasi serial. Format RTU dilengkapi dengan mekanisme *cyclic redundancy error* (CRC) untuk memastikan keandalan data. Modbus RTU merupakan implementasi protokol modbus yang paling umum digunakan. Setiap *frame* data dipisahkan dengan periode *idle* (silent).

3.2.3 Perancangan Modul Rangkaian RTC DS1307

Pada Tugas Akhir ini kami membutuhkan *update* waktu terkini pada kedua relai yang berguna pada saat *user* ingin melihat kondisi kedua relai tersebut pada waktu tertentu. Untuk dapat mencatat waktu terkini dibutuhkan perhitungan waktu yang akurat. Perhitugan waktu ini dimulai dari tahun, bulan, tanggal, jam, menit, dan detik untuk perhitungan yang akurat. Perhitungan tadi akan diolah oleh sebuah modul *Real Time Clock* (RTC) yang berfungsi untuk *update* waktunya. Tugas akhir kami ini menggunakan RTC berjenis DS1307. Pada modul ini sudah memiliki sumber sendiri menggunakan baterai, jadi pada saat relai *trip* maka waktu akan tetap *update*.

Modul RTC DS1307 dapat langsung dihubungkan pada arduino. Sambungan dengan Arduino Mega dapat dituliskan sebagai berikut :

1. VCC – pin 5V Arduino Mega
2. GND – pin *Ground* Arduino Mega
3. SDA – pin 20 (SDA) pada Arduino Mega
4. SCL – pin 21 (SDL) pada Arduino Mega



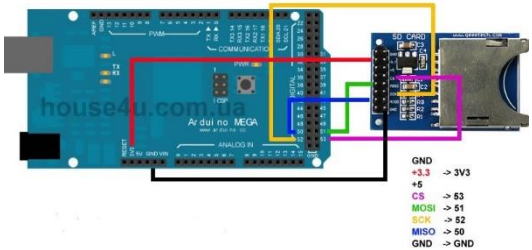
Gambar 3.6 *Wiring* modul RTC DS1307 dengan Arduino Mega

3.2.4 Perancangan Modul Rangkaian *micro*SD *Card*

Selain menggunakan RTC untuk *update* waktu terkini, pada tugas akhir ini kami juga membutuhkan simpanan data terkini maupun yang lama dari keadaan dari kedua relai. Simpanan data ini berisi diantaranya arus dan tegangan yang tercatat pada kedua relai, jumlah *trip* serta waktu terkini saat kedua relai mengalami *trip*, penyebab kedua relai mengalami *trip*. Jadi untuk menyimpan data – data terkini maupun terdahulu yang terjadi pada kedua relai ini atau bisa disebut *data logger* dibutuhkan sebuah wadah yaitu *micro*SD *Card*.

*micro*SD *Card* ini akan terpasang pada modul *microSD Adapter* agar data – data yang dibutuhkan bisa disimpan pada *micro*SD *Card*. Berikut adalah sambungan antara *microSD Adapter* dengan Arduino Mega :

1. CS (*Chip Select*) – Pin 53 (SS) Arduino Mega
2. SCK (Serial *Clock*) – Pin 52 (SCK) Arduino Mega
3. MISO (Serial Data *Out*) – Pin 50 (MISO) Arduino Mega
4. MOSI (Serial Data *In*) – Pin 51 (MOSI) Arduino Mega
5. VCC – Pin 5 Volt Arduino Mega
6. GND – Pin *Ground* Arduino Mega



Gambar 3.7 *Wiring* modul *micro*SD *Card* dengan Arduino Mega

3.3 Perancangan Perangkat Lunak (*Software)*

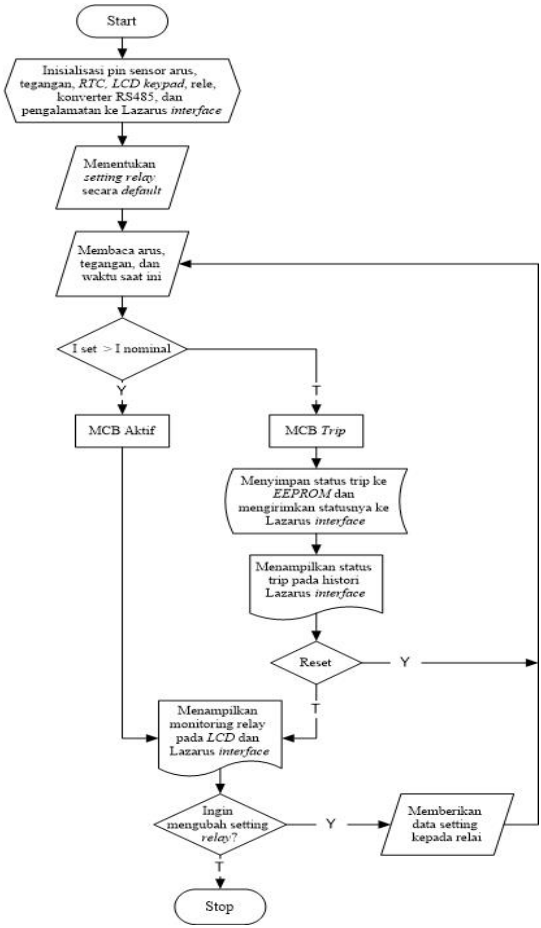
Perancangan perangkat lunak (*software*) menggunakan dua macam *software* yakni Arduino IDE untuk pemrograman pada papan Arduino serta menjalankan fungsi dari sensor dan perangkat elektronik yang terhubung pada Arduino, sedangkan software Lazarus digunakan untuk merancang pada sisi *interface* dengan PC yang nantinya akan menampilkan dan memberikan *setting* pada relai menggunakan protokol Modbus RTU.

3.3.1 Pemrograman *Software* Arduino IDE

Software Arduino IDE pada Tugas Akhir digunakan untuk melakukan pemrograman papan Arduino dalam menjalankan sistem secara keseluruhan. Dan arduino yang kami gunakan adalah Arduino Mega. *Software* ini menggunakan bahasa pemrograman C. Perancangan dari pemrograman ini seperti yang ditunjukkan pada Gambar 3.8, dimana ketika program pertama kali dijalankan akan menyimpan *setting default* dari relai yang meliputi *setting* arus, waktu, dan tipe karakteristik relai yang digunakan. Selanjutnya relai akan membaca besar arus, besar tegangan yang terdapat dalam jaringan dan juga waktu yang dibaca oleh RTC. Jika arus nominal jaringan kurang dari arus *setting*, maka MCB relai utama akan terhubung. Namun jika arus nominal lebih besar dari arus *setting* maka relai akan mendeteksi gangguan arus lebih dan MCB akan *trip* atau memutuskan jaringan sesuai dengan waktu dan tipe karakteristik relai yang digunakan.

Pada saat itu juga status trip pada relai akan disimpan di EEPROM pada arduino mega yang kapasitasnya 4 kb. EEPROM ini adalah sejenis chip memori tidak-terhapus yang digunakan dalam komputer dan peralatan elektronik lain (seperti Arduino mega) untuk menyimpan sejumlah konfigurasi data pada alat elektronik tersebut yang tetap harus terjaga meskipun sumber daya diputuskan. Setelah disimpan, status trip akan dikirimkan ke *interface* lazarus sebagai data logger yang akan menampilkan histori trip dari relai. Tetapi apabila tombol reset relai ditekan maka status trip akan menjadi status normal kembali dan relai akan mulai membaca arus pada jaringan lagi. Data yang ada pada relai akan ditampilkan pada LCD, disimpan pada data logger dan dikirimkan ke lazarus sebagai *interface* di komputer. Unutk LCD akan menampilkan arus *setting*, arus nominal dan tegangan, waktu saat ini, dan juga status relai.

Dan untuk lazarus disini menampilkan arus *setting*, arus nominal, status relai, waktu *setting trip*, waktu *countdown trip*, waktu saat ini, dan juga tipe karakteristik yang digunakan pada kedua relai. Jika *setting* pada relai ingin diganti, maka bisa di ganti lewat lazarus ini, dengan cara mengubah *setting* yang diperlukan lalu dimasukkan ke dalam kolom lazarus dan dikirim ke relai yang perlu di *setting* ulang. Untuk lebih ringkasnya dapat dilihat pada *flowchart* pemrograman Arduino IDE di bawah ini.



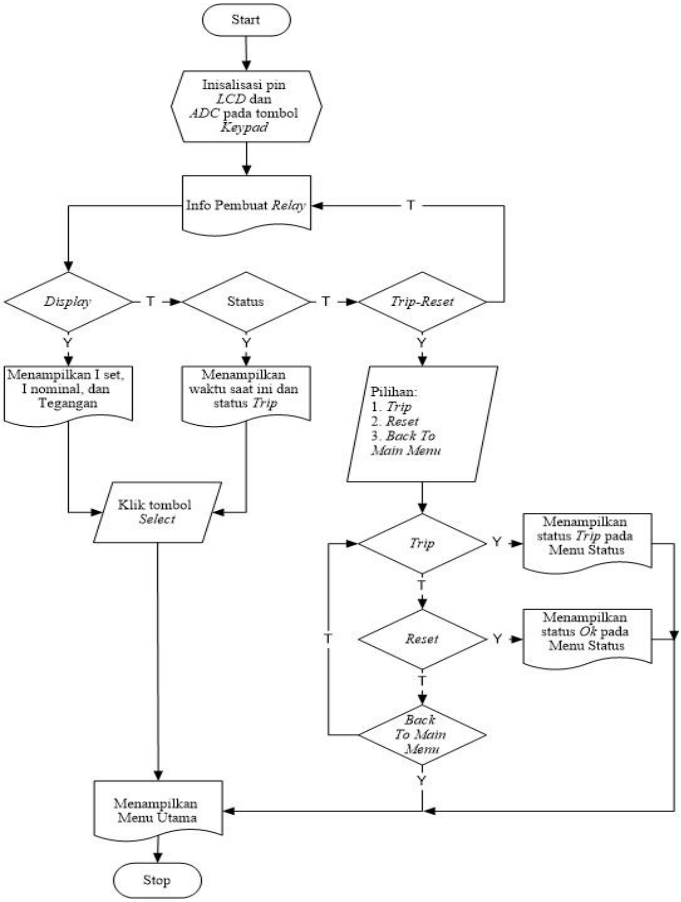
Gambar 3.8 *Flowchart* Pemrograman *Softwarre* Arduino IDE

3.3.2 Pemrograman LCD *Keypad Shield*

Untuk pemrograman *Liquid Crystal Display* (LCD) pada tugas akhir ini digunakan untuk mempermudah *user* untuk melihat kondisi terkini dari relai. Pada pemrograman *software* Arduino IDE telah tersedia *library* untuk pemrograman LCD itu sendiri. Sedangkan untuk *keypad* dan menu yang tampil pada LCD kami menggunakan program tersendiri untuk melengkapi program yang telah ada.

Agar LCD *Keypad* ini dapat digunakan sebagai penampil menu kondisi yang terjadi pada relai maka dibutuhkan program yang sesuai untuk LCD *Keypad* ini. Untuk urutan cara kerja dari *flowchart* adalah sebagai berikut :

1. *Start* adalah ketika program dimulai.
2. Untuk *wiring* LCD *Keypad* ini tombol dihubungkan pada Analog 0 Arduino, DB 4 pada Digital 4, DB5 pada digital 5, DB 6 pada Digital 6, DB7 pada Digital 7, pin RS pada Digital 8, *Enable* pada Digital 9 dan *Backlight Control* pada pin Digital 10.
3. Selanjutnya inisialisasi ADC yang akan digunakan untuk mengatur tombol pada LCD *Keypad*.
4. Tombol yang digunakan adalah tombol *Down* untuk menampilkan pilihan menu dan tombol *Select* untuk memilih menu.
5. Menu yang akan ditampilkan ada 3, yaitu :
   a. Menu *Display* yang berisi I *setting Short Circuit* dan *Over Load*, I rms dan V rms.
   b. Menu Status yang berisi waktu terkini dari RTC dan status *trip* relai.
   c. Menu *Test-Trip* yang berisi *Test Trip*, *Reset* dan pilihan untuk kembali ke Menu Awal.
6. *Stop* adalah ketika program berhenti.



Gambar 3.9 *Flowchart* Pemrograman LCD *Keypad Shield*

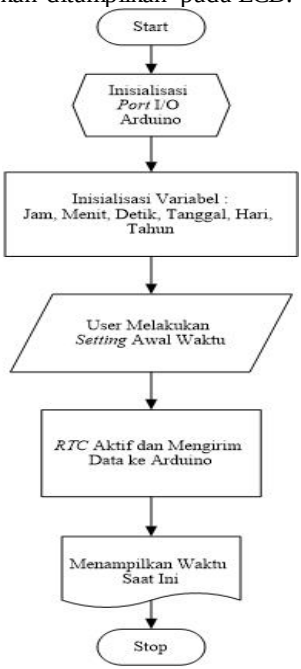
3.3.3 Pemrograman RTC

Untuk jenis *Real Time Clock* (RTC) yang digunakan pada Tugas Akhir ini, yaitu Modul *Real Time Clock* DS1307. Pemrograman pada papan Arduino telah tersedia *library* dari RTC tersebut. *Wiring* RTC ke Arduino dengan I_2C, yaitu menghubungkan pin SDA dan SCL dari RTC ke Arduino pada pin 20 dan 21.

Pengguna dapat memanggil *library* yang telah tersedia pada Arduino IDE. Agar RTC ini dapat digunakan sebagai pemberi data waktu

dengan baik maka dibutuhkan program yang sesuai untuk RTC ini. Pada RTC ini set awal untuk data hari, tanggal, serta waktu diberikan pada program dan tidak di set secara manual setelah program di*upload*. Untuk urutan cara kerja dari *flowchart* adalah sebagai berikut :

1. *Start* adalah ketika program dimulai.
2. Untuk *wiring* RTC dengan Arduino dihubungkan pada pin SDA dan SCL jadi pada inisialisasi *port* I/O Arduino harus mengaktifkan pin SDA dan SCL. Yakni dengan mengaktifkan komunikasi I_2C.
3. Selanjutnya inisialisasi variabel yang akan digunakan untuk detik, menit, jam, tanggal, dan hari.
4. Selanjutnya *user* akan melakukan *setting* awal untuk waktu pada RTC tersebut.
5. Dengan data *setting* awal tersebut RTC akan mengirim data tersebut ke Arduino dengan komunikasi I_2C yakni melalui pin SDA dan SCL.
6. Data waktu akan ditampilkan pada LCD.



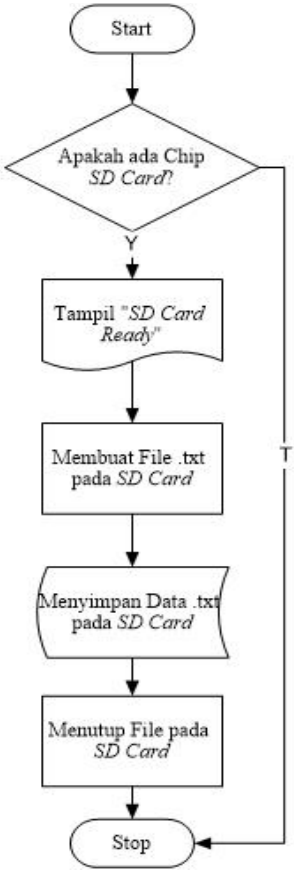
Gambar 3.10 *Flowchart* Pemrograman RTC

3.3.4 Pemrograman SD *Card*

Untuk pemrograman pada SD *Card* ini digunakan sebagai data *logger* dari kedua relai. Data *logger* ini untuk menyimpan data – data terkini maupun terdahulu yang terjadi pada kedua relai. Komunikasi data *logger* berbeda dengan komunikasi dengan RTC yaitu menggunakan SPI. Lebih jelasnya menggunakan pin MISO, MOSI, SCK, dan CS. Pada rancangan *hardware* pin CS terletak pada pin I/O 53. Pin tersebut akan digunakan sebagai acuan untuk mengaktifkan komunikasi dengan relai. Agar *SD Card* ini berjalan dengan baik pada Arduino, maka dibutuhkan pemrograman yang sesuai dengan kebutuhan dari *SD Card*. Untuk urutan cara kerja dari *flowchart* adalah sebagai berikut :

1. *Start* adalah ketika program dimulai.
2. Dilakukan pengecekan untuk mengetahui ada atau tidaknya *chip SD Card*.
3. Jika terdapat *chip SD Card* maka akan dimulai untuk proses penyimpanan dari data yang akan disimpan, sedangkan jika tidak ada *chip SD Card* maka program akan berhenti.
4. Tampilan "SD *Card Ready*" untuk proses selanjutnya.
5. Pada tahap ini akan dimulai proses penyimpanan data yang diawali dengan membuka *file* pada *SD Card*.
6. Data akan tersimpan pada *file* yang telah dibuka pada tahap sebelumnya.
7. Setelah data tersimpan, selanjutnya *file* akan ditutup dan data telah selesai tersimpan

Program tersebut digunakan untuk menulis data *String* pada *file*. Setiap pengaksesan *SD Card* dimulai dengan perintah SD.*open*(); untuk menulis data yang tersimpan dalam memori pada *file* menggunakan dataFile.*print* (); ketika penulisan selesai maka akan ditutup dengan dataFile.*close*();.



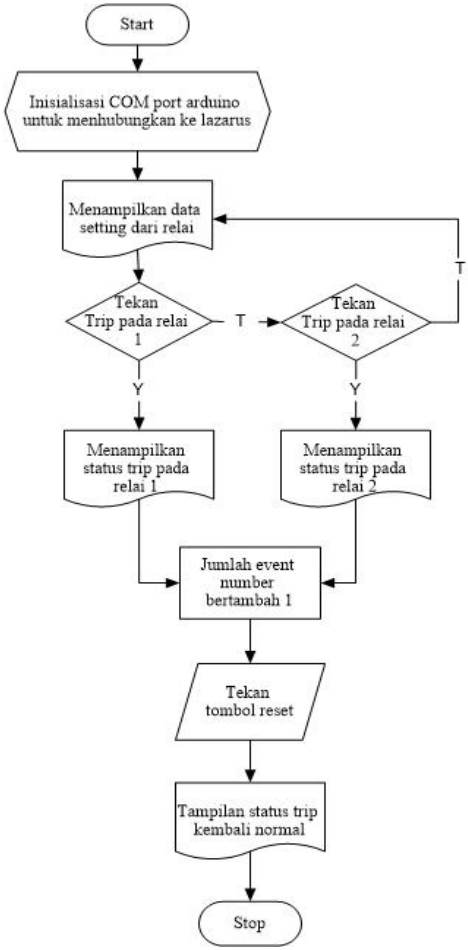
Gambar 3.11 *Flowchart* Pemrograman SD *Card*

3.3.5 Pemrograman *Software* Lazarus

Software Lazarus ini digunakan sebagai perancangan *interface* pada komputer yang berfungsi untuk mengamati besar arus nominal dan *setting* dari kedua relai dan juga bisa meng-*setting* langsung ke relainya sesuai keperluan. Pada tampilan Lazarus terdapat 3 *tab* antara lain, *Monitoring*, *Configuration*, dan *COM Setting*. Untuk pilihan yang atas ada *File* yang jika dipilih hanya ada pilihan *exit* yang berfungsi untuk keluar dari aplikasi Lazarus tersebut. Ada *Historical* yang jika dipilih hanya ada pilihan *Historical Data* yang berfungsi untuk menampilkan riwayat

kejadian trip yang dialami oleh relai. Yang ditampilkan pada *historical data* ini antara lain, data *setting* relai, waktu pada saat relai trip, arus nominal dan tegangan, arus *fault*, penyebab relai mengalami *trip*, dan jumlah relai mengalami *trip*. Selanjutnya ada *Help* yang jika dipilih ada dua pilihan yaitu *Trouble Shooting* dan *About*. Untuk *"Trouble Shooting"* ini berisi tentang prosedur untuk menggunakan aplikasi Lazarus tersebut. Dan untuk *"About"* ini hanya berisi info tentang aplikasi Lazarus ini seperti versinya, kegunaan dibuatnya aplikasi ini, protokol yang digunakan untuk komunikasi data dengan relai, dan profil dari pembuat aplikasi relai ini yaitu kami.

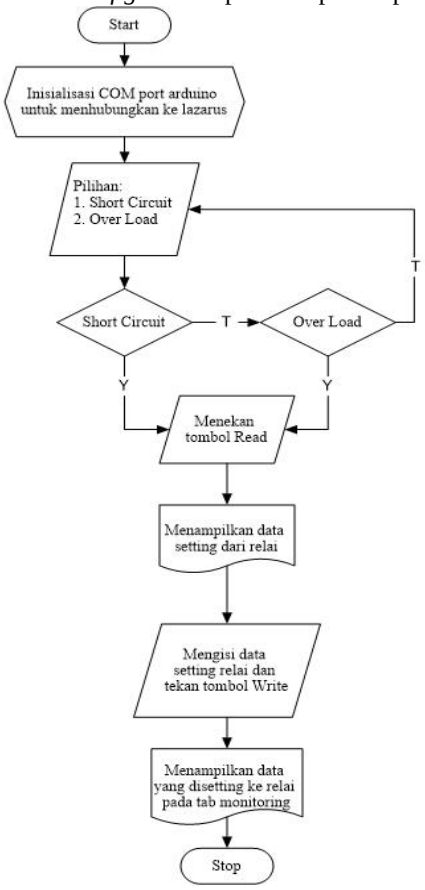
Pada *tab COM Setting* ini hanya untuk menyambungkan antara Lazarus dengan Arduino agar bisa saling berkomunikasi. Lalu pada *tab Monitoring* ini berfungsi unutuk menampilkan semua data *setting* yang ada pada kedua relai dengan 2 kategori yaitu *Short Circuit* dan *Over Load*. Data *setting* ini berisi antar lain, arus *setting*, arus nominal, status *trip* relai, tipe karakteristik yang diigunakan pada relai, waktu *setting trip*, waktu *countdown trip*, jumlah *trip* yang dialami relai dan 3 perintah. 3 perintah ini antara lain, *Clear Event*, *Trip* dan *Reset*. Perintah *"Clear Event"* ini berfungsi untuk memulai kembali jumlah *trip* yang dialami relai menjadi 0. Untuk kedua perintah selanjutnya sebenarnya berhubungan satu sama lain yaitu ketika pilihan *"Trip"* dipilih dan menekan tombol *"Send"* maka kontaktor akan langsung ikut *trip* sesuai dengan perintah, jika pilihan *"Reset"* dipilih dan menekan tombol *"Send"* maka relai akan me-*reset* dirinya sendiri untuk kembali normal dan kontaktor juga kembali normal kembali. Berikut adalah *flowchart* dari *tab monitoring* pada tampilan aplikasi Lazarus.



Gambar 3.12 *Flowchart* Pemrograman Lazarus pada *tab Monitoring*

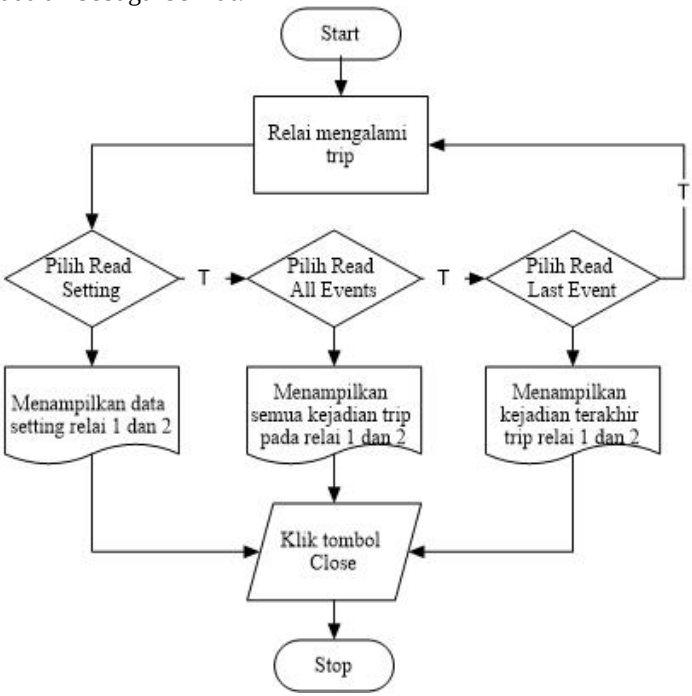
Pada *tab Configuration* ini berfungsi unutuk memberikan data *setting* pada kedua relai dengan 2 kategori *setting* yaitu *setting* pada *Short Circuit* dan *Over Load*. Data *setting* ini berisi antar lain, tipe kategori *setting* yang dipilih antara *short circuit* dan *over load*, arus *setting*, tipe karakteristik yang akan diigunakan pada relai, waktu *setting trip*, *Time Multiplier Setting* (TMS) untuk menggunakan tipe karakteristik inverse,

dan 3 perintah. 3 perintah ini antara lain, *Set Default*, *Read* dan *Write*. Perintah *"Set Default"* ini berfungsi untuk memberikan *setting default* yang digunakan pada relai saat pertama kali diprogram. Untuk kedua perintah selanjutnya sebenarnya berhubungan satu sama lain yaitu ketika menekan tombol pilihan *"Read"* maka aplikasi Lazarus ini akan membaca data *setting* pada relai dan ditampilkan pada *tab configuration*. Jika tombol pilihan *"Write"* ditekan maka relai akan mengirim data *setting* yang kita atur pada *tab configuration* ini kepada relai. Berikut adalah *flowchart* dari *tab configuration* pada tampilan aplikasi Lazarus.



Gambar 3.13 *Flowchart* Pemrograman Lazarus pada *tab Configuration*

Yang terakhir ini akan membahas tentang *Historical Data* yang sudah dijelaskan sebelumnya yaitu berfungsi untuk menampilkan riwayat kejadian trip yang dialami oleh relai. Jadi unutk *flowchart* dari *historical data* adalah sebagai berikut.



Gambar 3.14 *Flowchart* Pemrograman Lazarus pada *tab Historical Data*

-----Halaman ini sengaja dikosongkan-----

BAB IV
PENGUJIAN DAN ANALISA DATA

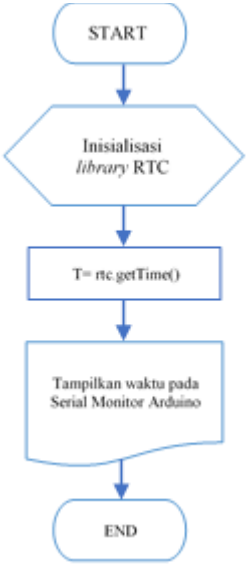
Pada bab 4 ini membahas mengenai pengujian dan analisa data dari setiap komponen utama maupun pendukung yang dibuat agar kinerja sistem dari alat tugas akhir kami dapat berjalan dengan baik sesuai dengan yang diharapkan dan jika ada masalah bisa mudah untuk mengetahui masalah tersebut lalu dapat cepat diselesaikan. Pengujian merupakan salah satu langkah yang harus dilakukan untuk mengetahui apakah sistem yang telah dibuat sesuai dengan yang direncanakan. Kesesuaian sistem dengan perencanaan dapat dilihat dari hasil-hasil yang dicapai pada pengujian sistem. Pengujian juga bertujuan untuk mengetahui kelebihan dan kekurangan dari sistem yang telah dibuat. Hasil pengujian tersebut akan dianalisa untuk mengetahui penyebab terjadinya kekurangan atau kesalahan dalam sistem

Pada bab 4 ini pengujian dan analisa data *hardware* dan *software* dari alat yang telah kami buat terdapat cukup banyak komponen yang harus duji dan dianalisa. Untuk bagian saya yang akan diuji adalah :

1. Pengujian RTC
2. Pengujian Memori SD *Card*
3. Pengujian LCD *Keypad Shield*
4. Pengujian Komunikasi RS485 dan *Software* Lazarus
5. Pengujian Alat Keseluruhan

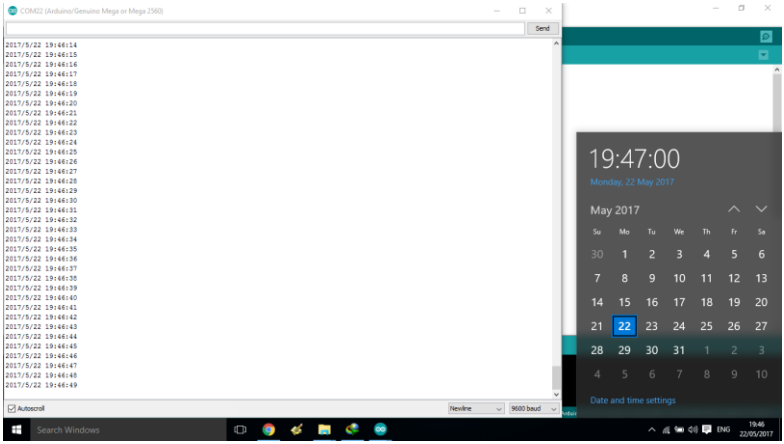
4.1 Pengujian RTC

Untuk menampilkan waktu terkini dari RTC ini dilakukan langsung dengan membaca data pada RTC meliputi data – data dari tahun, bulan, hari, jam, menit, dan detik waktu terkini. Sebelum dilakukan pengujian pada RTC maka Arduino Mega diberi program sesuai dengan rancangan *flowchart* pada gambar 4.1 berikut ini.

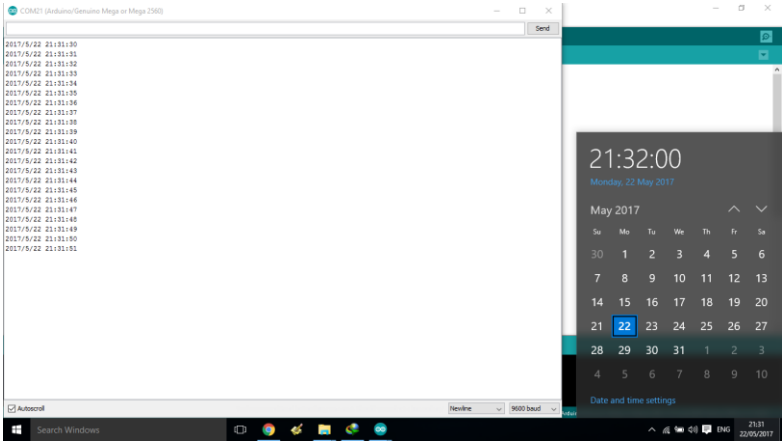


Gambar 4.1 *Flowchart* Pengujian RTC

Setelah program berhasil *upload* ke Arduino Mega, langkah selanjutnya adalah menghubungkan antara Arduino Mega dengan RTC dengan *wiring* seperti pada Gambar 3.6. Setelah itu baru dilakukan pengujian pada RTC. Pengujian ini dilakukan dengan cara membandingkan tampilan jam pada serial monitor Arduino IDE yang diambil dari data RTC dengan tampilan jam pada komputer kami. Salah satu gambar dari pengujian RTC pada kedua relai ditunjukkan pada Gambar 4.2 dan Gambar 4.3 di bawah ini.



Gambar 4.2 Pengujian RTC Relai 1 dengan Waktu pada Komputer



Gambar 4.3 Pengujian RTC Relai 2 dengan Waktu pada Komputer

Dikarenakan pada tugas akhir ini kami membuat 2 relai, maka pengujian ini juga dibedakan menjadi 2 macam, yaitu pengujian RTC pada relai 1 dan relai 2. Hasil pengamatan RTC yang sudah dibandingkan dengan tampilan jam yang terdapat pada komputer ditunjukkan pada table di bawah ini.

Tabel 4.1 Pengujian RTC pada Relai 1

NO	YANG DIUJI	TAMPILAN SERIAL MONITOR	TAMPILAN KOMPUTER	SELISIH
1	WAKTU	19:46:47	19:47:00	00:00:13
2		19:49:50	19:50:01	00:00:11
3		19:52:48	19:53:00	00:00:12
4		19:55:49	19:56:00	00:00:11
5		19:58:48	19:59:00	00:00:12
6		20:01:48	20:02:00	00:00:12
7	TANGGAL	2017/05/22	22 Mei 2017	-

Tabel 4.2 Pengujian RTC pada Relai 2

NO	YANG DIUJI	TAMPILAN SERIAL MONITOR	TAMPILAN KOMPUTER	SELISIH
1	WAKTU	21:31:51	21:32:00	00:00:09
2		21:34:51	21:35:00	00:00:09
3		21:37:51	21:38:00	00:00:09
4		21:40:51	21:41:00	00:00:09
5		21:43:51	21:44:00	00:00:09
6		21:46:50	21:47:00	00:00:10
7	TANGGAL	2017/05/22	22 Mei 2017	-

Pada Tabel 4.1 dan 4.2 ditunjukkan hasil pengujian perbandingan dari RTC dengan tampilan jam pada komputer. Hasil pengujian ini menunjukkan waktu terkini yang terdiri dari tahun, bulan, hari, jam, menit dan detik pada RTC telah sesuai dengan waktu terkini yang tampil pada komputer seperti pada Gambar 4.2 dan Gambar 4.3. Pengujian ini dilakukan sebanyak 6 kali pengambilan data per detik dengan selisih rata – rata atau nilai *error* 10 detik, maka dapat disimpulkan bahwa RTC telah sesuai dan dapat digunakan pada tugas akhir kami.

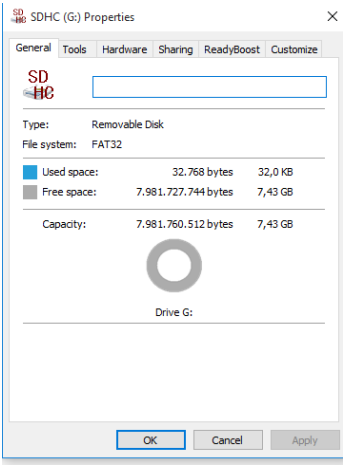
4.2 Pengujian Memori SD *Card*

SD *Card* dalam tugas akhir kami ini berfungsi utuk menyimpan semua kejadian terkini maupun terdahulu yang dialami oleh kedua relai. Pada pengujian SD *Card* ini dilakukan untuk mengetahui kapasitas yang dapat ditampung oleh SD *Card*. Untuk kapasitas memori yang digunakan adalah *Micro* SD *Card* VGen dengan kapasitas penyimpanan 8 Gb.

Pengujian ini dilakukan dengan cara pembacaan kapasitas kartu pada komputer dalam kondisi kosong. Pengujian ini dilakukan untuk memastikan SD *Card* memiliki ruang penyimpanan yang benar – benar kosong dan kapasitas yang sama dengan yang diharapkan. Memori ini nantinya akan diisi data arus dan tegangan. Berikut adalah data SD *Card* yang telah diambil dan terbagi menjadi beberapa bagian yang seperti pada Tabel 4.3 dan 4.4.

Tabel 4.3 Pengujian SD *Card* Relai 1

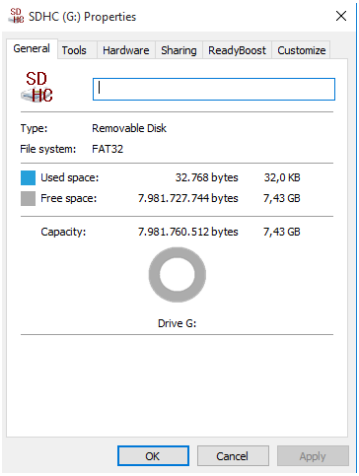
Nama Ruang	**Ukuran (*bytes*)**	**Ukuran**
Used Space	32.768	32.0 KB
Free Space	7.981.727.744	7.43 GB
Capacity	7.981.760.512	7.43 GB



Gambar 4.4 Hasil Pengujian SD *Card* Relai 1

Tabel 4.4 Pengujian SD *Card* Relai 2

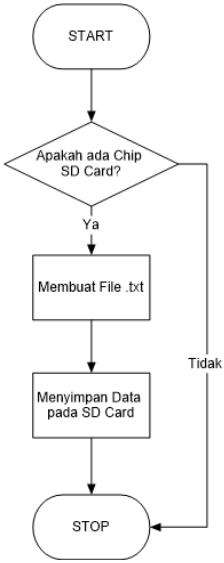
Nama Ruang	Ukuran (*bytes*)	Ukuran
Used Space	32.768	32.0 KB
Free Space	7.981.727.744	7.43 GB
Capacity	7.981.760.512	7.43 GB



Gambar 4.5 Hasil Pengujian SD *Card* Relai 2

Dengan demikian maka kondisi SD *Card* dapat digunakan pada Tugas Akhir ini untuk menyimpan nilai arus dan tegangan pada segala kondisi yang dialami relai. Selain dengan mengecek memori kartu, dilakukan juga pengujian penyimpanan data arus dan tegangan ke dalam *file text* (.txt). Gambar 4.6 adalah gambar *flowchart* untuk penyimpanan data pada *file text* (.txt). Agar file dapat dijadikan sebuah data yang dapat diproses, maka file .txt diubah kedalam file .xlxs (Microsoft Excel). Langkah langkah untuk mengubah data .txt menjadi .xlxs adalah sebagai berikut :

1. Buka Microsoft Excel
2. Buka menu Data
3. Pada Submenu "*Get External Data*", pilih "*From Text*"
4. Selanjutnya pilih file .txt penyimpanan data dari Arduino
5. Akan muncul jendela baru "*Text Import Wizard*", pilih *Delimited* lalu *Next*
6. Lalu pilih *Delimited* (pemisah data) yang digunakan pada *file* .txt, lalu pilih *Next*
7. Pada data *format* pilih *General*. Tekan *Finish*
8. Pada jendela Import Data centang "*Add this data to the Data Model*" lalu klik OK



Gambar 4.6 *Flowchart* Penyimpanan Data .txt pada SD *Card*

Pengambilan data penyimpanan data dilakukan pada kedua relai, yaitu relai 1 dan relai 2. Pada kedua relai ini akan dilakukan seperti langkah – langkah di atas sebagai bukti bahwa Memori SD *Card* dapat berfungsi dengan baik dan benar. Tabel 4.5 dan 4.6 menunjukkan data yang telah di *import* ke dalam Microsoft Excel.

Tabel 4.5 Penyimpanan Data SD *Card* Relai 1

No	Tanggal	Waktu	Arus(A)	Tegangan(V)
1	27/05/2017	22:20:04	0,03	209,54
2	27/05/2017	22:20:06	0,03	209,1
3	27/05/2017	22:20:07	0,03	210,09
4	27/05/2017	22:20:09	0,03	209,65
5	27/05/2017	22:20:10	0,03	208,96
6	27/05/2017	22:20:12	0,03	209,75
7	27/05/2017	22:20:13	0,03	209,78
8	27/05/2017	22:20:15	0,04	209,58
9	27/05/2017	22:20:16	0,03	209,29
10	27/05/2017	22:20:17	0,03	209,2
11	27/05/2017	22:20:19	0,04	209,29
12	27/05/2017	22:20:20	0,03	209,28
13	27/05/2017	22:20:22	0,03	209,14
14	27/05/2017	22:20:23	0,03	208,82
15	27/05/2017	22:20:25	0,03	208,82

Pada Gambar 4.7 berikut ini yang menampilkan data arus dan tegangan pada LCD, dan telah disimpan di SD Card seperti yang ditunjukkan pada tabel 4.5 di atas.

Gambar 4.7 Tampilan LCD Relai 1

Tabel 4.6 Penyimpanan Data SD *Card* Relai 2

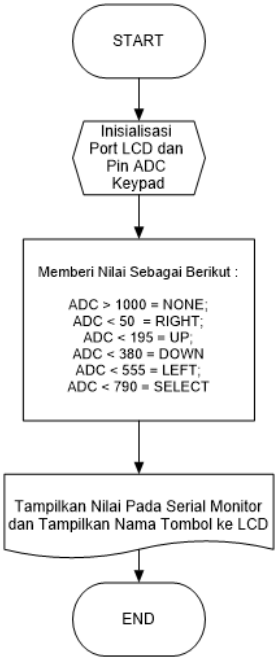
No	Tanggal	Waktu	Arus(A)	Tegangan(V)
1	27/05/2017	22:19:03	0,09	60,9
2	27/05/2017	22:19:43	0,06	233,24
3	27/05/2017	22:19:46	0,04	195,49
4	27/05/2017	22:19:48	0,03	193,93
5	27/05/2017	22:19:49	0,03	195,1
6	27/05/2017	22:19:51	0,03	195,15
7	27/05/2017	22:19:52	0,23	194,66
8	27/05/2017	22:19:54	0,25	194,79
9	27/05/2017	22:19:55	0,24	194,52
10	27/05/2017	22:19:56	0,25	194,56
11	27/05/2017	22:19:58	0,24	194,25
12	27/05/2017	22:19:59	0,66	192,9
13	27/05/2017	22:20:01	0,65	192,96
14	27/05/2017	22:20:02	0,66	192,81
15	27/05/2017	22:20:04	0,65	193

Pada Gambar 4.8 berikut ini menampilkan data arus dan tegangan pada LCD, dan telah disimpan di SD Card seperti yang ditunjukkan pada tabel 4.6 di atas.

Gambar 4.8 Tampilan LCD Relai 2

4.3 Pengujian LCD *Keypad Shield*

Pada pengujian LCD *Keypad Shield* ini dilakukan dengan cara menambahkan program pada mikrokontroler Arduino untuk menampilkan karakter pada LCD dan membaca ADC dari tombol *Keypad* ketika ditekan. Pada pengujian LCD kali ini menggunakan LCD *Keypad* 16 x 2, pengujian kami lakukan dengan memberi program untuk menampilkan nama tombol *Keypad* yang telah ditekan pada LCD dan menapilkan ADC pada serial monitor. Pada Gambar 4.9 berikut ini adalah *flowchart* pemrograman yang digunakan untuk pengujian LCD pada Relai 1 dan Relai 2.



Gambar 4.9 *Flowchart* Pengujian LCD

Setelah program berhasil *upload* ke *board* Arduino Mega, selanjutnya melakukan penyambungan antara Arduino Mega dengan LCD *Keypad Shield*. Pada Tabel 2.1 telah dijelaskan konfigurasi penyambungan antara Arduino Mega dengan LCD *Keypad Shield*. Tabel 4.7 dan 4.8 berikut ini adalah tabel hasil pengujian LCD *Keypad* 16x2 yang dipasang pada Relai 1 dan Relai 2.

Tabel 4.7 Pengujian LCD *Keypad Shield* pada Relai 1

No	Nilai pada Program	Tampilan LCD	ADC pada Serial Monitor
1	ADC > 1000 NONE	NONE	1023
2	ADC < 50 RIGHT	RIGHT	0
3	ADC < 195 UP	UP	99
4	ADC < 380 DOWN	DOWN	255
5	ADC < 555 LEFT	LEFT	407
6	ADC < 790 SELECT	SELECT	638

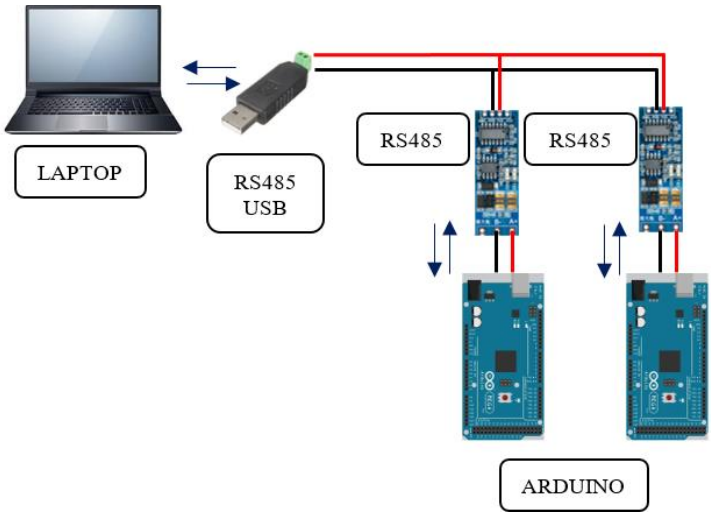
Tabel 4.8 Pengujian LCD *Keypad Shield* pada Relai 2

No	Nilai pada Program	Tampilan LCD	ADC pada Serial Monitor
1	ADC > 1000 NONE	NONE	1023
2	ADC < 50 RIGHT	RIGHT	0
3	ADC < 195 UP	UP	99
4	ADC < 380 DOWN	DOWN	255
5	ADC < 555 LEFT	LEFT	409
6	ADC < 790 SELECT	SELECT	639

Dari semua data pengujian di atas dapat dilihat bahwa antara tampilan pada LCD dan ADC yang tampil pada serial monitor Arduino IDE sesuai dengan nilai yang dimasukkan pada program Arduino. Oleh karena itu, dapat dikatakan bahwa LCD ini dapat berjalan dengan baik pada Relai 1 maupun Relai 2.

4.4 Pengujian Komunikasi RS485 dan *Software* Lazarus

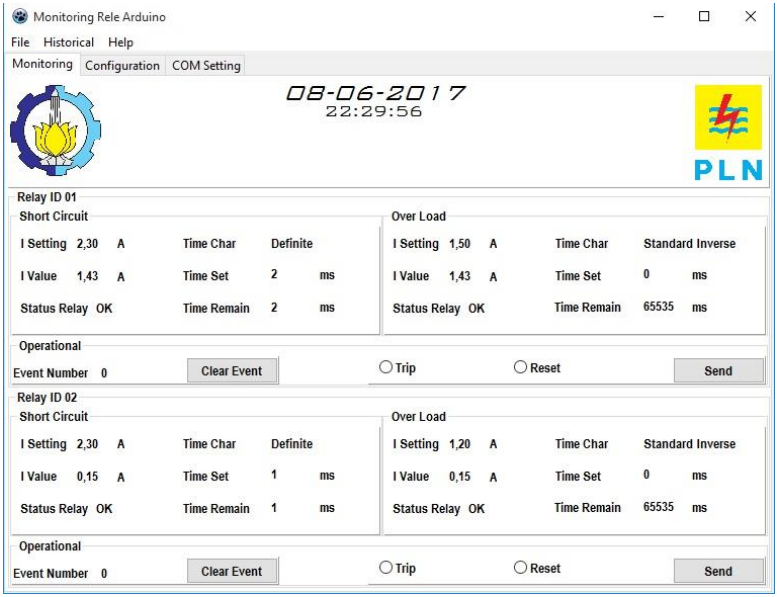
Pengujian komunikasi RS485 bertujuan untuk mengetahui apakah *master* dapat mengambil data pada *slave* dan juga mengirimkan data pada *slave*. Pengujian ini menggunakan *interface* yang kami gunakan pada komputer yaitu *software* Lazarus dengan menghubungkan ke kedua relai. Kedua arduino yang ada pada relai akan di *wiring* dengan modul RS485, begitu juga dengan komputer yang digunakan juga disambungkan ke modul RS485 dan ground dari sambungan semuanya dijadikan satu. Untuk sistem *wiring* dari modul RS485 bisa dilihat pada gambar di bawah ini.



Gambar 4.10 *Wiring* Kedua Relai dengan Komputer lewat RS485

Untuk sistem *monitoring* dan *setting* ini bisa juga disebut sebagai sistem *Supervisory Control And Data Acquisition* (SCADA). Untuk komunikasi data pada sistem SCADA ini yaitu komunikasi antar kedua relai (slave) dengan Lazarus (master) ini kami menggunakan protokol Modbus RTU. Disini *master* berfungsi untuk aktif mengambil data

monitoring dari relai pada saat pertama kali disambungkan. Jadi saat pertama kali Lazarus (master) membaca COM dari RS485, monitoring yang ada pada Lazarus akan langsung *update* sesuai dengan data – data yang ada pada kedua relai yang berperan sebagai *slave*. Begitu pun juga saat Lazarus (master) memberikan *setting* pada relai (slave), maka kedua relai juga langsung *update* mengikuti dari data – data *setting* yang diberikan pada Lazarus dengan cara menampilkannya pada LCD yang ada pada setiap relai. Berikut ini adalah tampilan *monitoring* dari *interface* Lazarus.



Gambar 4.11 *Monitoring* Relai pada Lazarus

Pada gambar di atas menunjukkan bahwa *monitoring* pada Lazarus yang sudah sesuai dengan data – data yang ada pada kedua relai ditunjukkan dengan tampilan pada LCD yang nilainya sama dengan tampilan pada Lazarus. Tampilan LCD bisa dilihat pada gambar berikut.

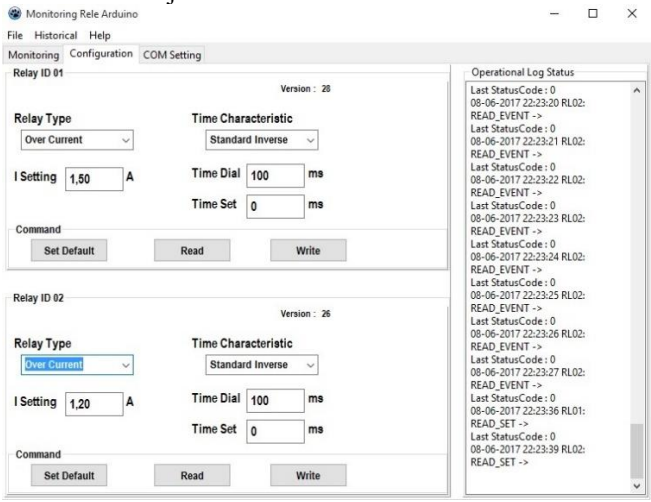
Gambar 4.12 Tampilan LCD pada Relia 1

Gambar 4.13 Tampilan LCD pada Relai 2

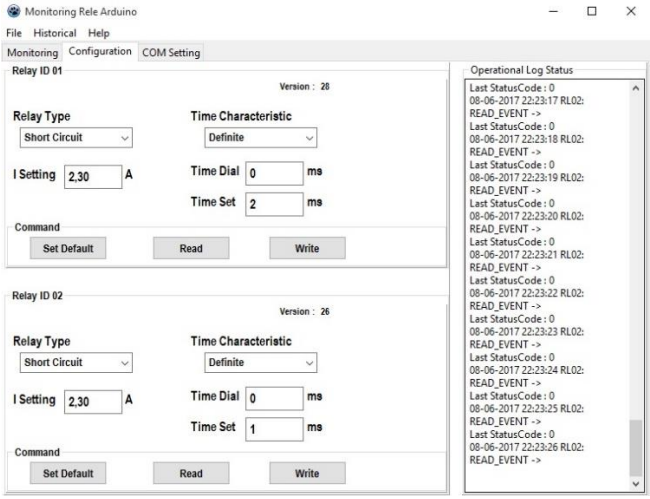
Berdasarkan gambar di atas *tab monitoring* yang ada pada Lazarus ini menampilkan antara lain, arus nominal, arus *setting*, status relai, tipe karakteristik yang digunakan, waktu *trip setting*, waktu *countdown trip*, jumlah *trip* yang dialami relai, perintah untuk memberikan *trip* dan *reset* pada relai. Untuk LCD relai 1 pada kolom "Set", terdapat 2 *setting* arus yaitu *setting over load* 1,50 A dan *setting short circuit* 2,30 A. Besar arus nominal pada LCD relai 1 terbaca 1,43 A. Bisa dilihat bahwa *setting* dan nilai arus nominal pada tampilan *monitoring* Lazarus dengan LCD pada relai 1 memang sama. Begitu pun juga pada LCD relai 2 dengan tampilan *monitoring* Lazarus. Untuk LCD relai 2 pada kolom "Set", terdapat 2

setting arus yaitu *setting over load* 1,20 A dan *setting short circuit* 2,30 A. Besar nominal pada LCD relai 2 terbaca 0,15 A.

Ada cara pengujian lainnya yaitu dengan memberikan perintah *trip* manual pada relai lewat Lazarus ini untuk membuktikan bahwa komunikasi antara *master* dengan *slave* sudah bisa terhubung melalui RS485. Untuk perintah manual ini yaitu dengan cara meng-*klik* pilihan *"Trip"* lalu meng-*klik* tombol *"Send"* yang ditunjukkan pada gambar di atas . Saat perintah *trip* dikirim ke relai maka kontaktor yang ada pada relai akan ikut *trip* dan status yang ada pada Lazarus dan juga LCD akan menampilkan tulisan status *trip* di karenakan "*remote"*. Yang dimaksudkan "*remote"* disini adalah Lazarus (master). Untuk mengembalikan relai menjadi normal kembali yaitu dengan meng-*klik* pilihan *"Reset"* lalu meng-*klik* tombol *"Send"* yang ada pada *tab monitoring* Lazarus pada gambar di atas. Saat relai sudah kembali normal maka status relai juga kembali normal kembali dengan *update* tulisan status relai menjadi "Ok" kembali.

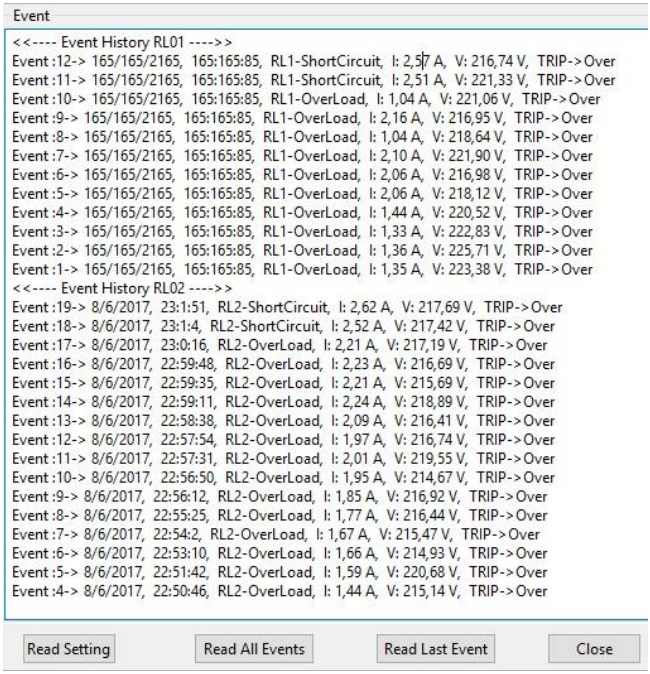


Gambar 4.14 *Setting Over Load* Relai pada Lazarus



Gambar 4.15 *Setting Short Circuit* pada Lazarus

Pada gambar di bawah ini menunjukkan *setting* Lazarus yang terletak pada *tab configuration* yang akan dikirim ke kedua relai. Pada gambar tersebut terdapat perintah *read* dan *write*. Untuk perintah *"read"* ini berfungsi untuk membaca data yang ada pada relai seperti pada *tab monitoring*. Sedangkan perintah *"write"* ini berfungsi untuk mengirimkan data *setting* dari Lazarus ke relai. Jadi pada saat *setting* yang baru sudah dikirim ke relai maka LCD yang ada pada relai tersebut akan berubah sesuai dengan data *setting* yang dikirimkan tadi. Setelah semua pengujian tadi sudah dilakukan dan didapatkan data yang sudah sesuai makan bisa disimpulkan bahwa pengujian komunikasi menggunakan RS485 sudah berhasil.



Gambar 4.16 Histori Relai pada Lazarus

Pada gambar di atas menunjukkan histori Lazarus. Histori ini berfungsi untuk menampilkan riwayat dari relai yang berisi tentang data *setting* relai, waktu pada saat relai trip, arus nominal dan tegangan, arus *fault*, penyebab relai mengalami *trip*, dan jumlah *trip* yang dialami relai. Jadi pada histori Lazarus ini kita bisa mengetahui riwayat semua kejadian *trip* yang terjadi pada kedua relai.

4.5 Pengujian Karakteristik *Over Load* dan *Short Circuit* Relai

Untuk mengetahui apakah *setting* yang telah di atur oleh Lazarus telah sesuai pada *setting* digital relay sesuai dengan yan diinginkan, maka di perlukan pengujian karakteristik waktu relai. Proses dalam pengambilan data yaitu relai akan di *setting* dengan nilai tertentu dan akan di beri beberapa pembebanan dan kombinasinya. Data ini berupa nilai arus dan waktu trip yang akan di olah dalam excel sehingga nilai dari

waktu *trip* perhitungan dan waktu *trip* pengukuraan dapat di bandingkan dalam bentuk grafik.

Dalam pengambilan data akan digunakan dua jenis kurva yaitu *standard inverse* dan *definite*. Untuk melakukan pengecekan pada kedua relai maka pengaturan Iset, Isc, waktu *trip* dan *time dial* di buat sama, dengan nilai masing masing;

1. Iset : 1.5 A
2. Isc : 2.3 A
3. Waktu *trip* : 0 s
4. *Time Dial* : 0.1 s

Sedangkan untuk pengujiannya di perlukan pembebanan, terdapat beberapa jenis dan kombinasi pembebanan agar arus yang terukur dapat di amati, adapun beban beban yang di gunakan sebagai berikut;

1. Setrika (350 watt)
2. Lampu (25 watt)
3. Lampu (100 watt)
4. Trip modul (440 watt)

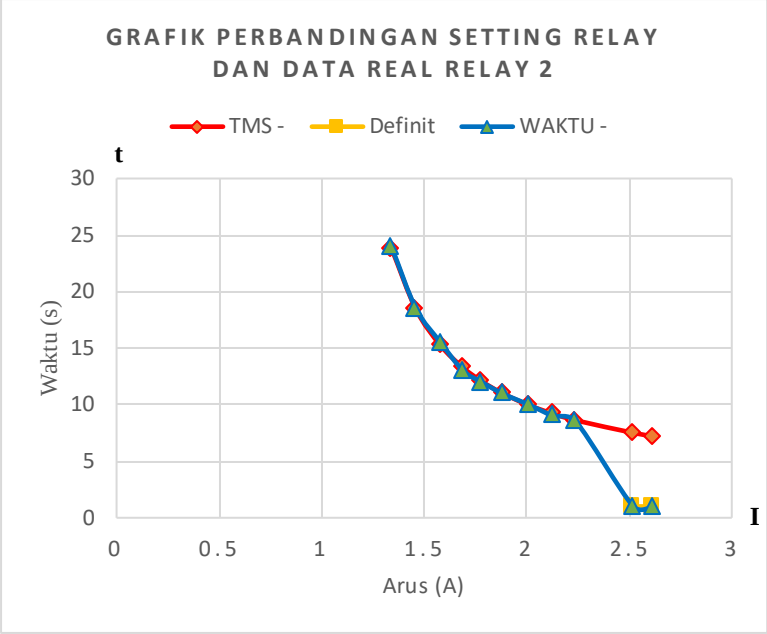
Tabel 4.9 berikut merupakan hasil dari data yang di dapat pada relay 1 dan 2 beserta perbandingan dengan perhitungan waktu trip.

Tabel 4.9 Hasil Pengujian pada Relai 2

SET RELAI 2 [OL = (Iset=1A ; Timedial=1s), SC =(Isc=2.3A ; Timeset= 0.5s)]							
NO	Beban		I NOM	Setting		Data Nyata	
	Beban 1	Beban 2		TMS	Definite	Waktu	Kondisi
1	lampu 200 watt		0.85	-		-	normal
2	rice cooker (cook) 300 watt		1.34	23.84782287		24	trip
3	rice cooker (cook) 300 watt	lampu 25 watt	1.46	18.42725015		18.5	trip
4	rice cooker (cook) 300 watt	lampu 50 watt	1.58	15.233167		15.5	trip

NO	Beban		I NOM	Setting		Data Nyata	
	Beban 1	Beban 2		TMS	Definite	Waktu	Kondisi
5	rice cooker (cook) 300 watt	lampu 75 watt	1.69	13.27035384		13	Trip
6	rice cooker (cook) 300 watt	lampu 100 watt	1.78	12.06998498		12	Trip
7	rice cooker (cook) 300 watt	lampu 125 watt	1.88	11.01887367		11	trip
8	rice cooker (cook) 300 watt	lampu 150 watt	2.01	9.95688092		10	trip
9	rice cooker (cook) 300 watt	lampu 175 watt	2.13	9.187941957		9	trip
10	rice cooker (cook) 300 watt	lampu 200 watt	2.24	8.609927145		8.5	trip
11	impedansi 440 watt	lampu 25 watt	2.52	7.503851129	1	1	trip
12	impedansi 440 watt	lampu 50 watt	2.62	7.19786038	1	1	trip

Berdasarkan tabel 4.9 di atas ditampilkan beberapa beban yang di kombinasikan akan membuat urutan arus yang semakin tinggi, yang memiliki beberapa karakter dimana arus yang bernilai di bawah 1 A tidak akan membuat *trip* relai 2, sedangkan arus yang bernilai di atas 1 A akan membuat relai 2 dalam kondisi *trip*. Waktu *trip* yang di peroleh dari nilai arus di atas 1 A dan di bawah 2.3 ampere memiliki waktu yang berbeda – beda. Semakin tinggi nilai arus pada nilai 1 sampai 2.3 A maka waktu *trip* semakin cepat, sedangkan waktu *trip* di atas 2.3 memiliki waktu yang sama. Untuk membuktikan apakah nilai dari data arus dan waktu *trip* sudah sesuai dengan kurva *setting* arus *standard inverse* dan *definite*, maka di perlukan pembandingan data dengan menggunakan grafik yang merupakan hasil dari perbadingan data Relai 2.



Gambar 4.17 Grafik Perbandingan *Setting* Relai dan Data *Real* Relai 2

Berdasarkan Gambar 4.17 grafik di atas nilai waktu *trip* relai 2 antara 1 A sampai dengan 2.3 A tidak terlalu jauh dari nilai *setting*, sehingga data yang di dapat sesuai dengan karakteristik dari kurva invers, sedangkan waktu *trip* relai gangguan yang lebih dari 2.3 A memiliki waktu kerja yang sama dengan karakteristik kurva *definite*.

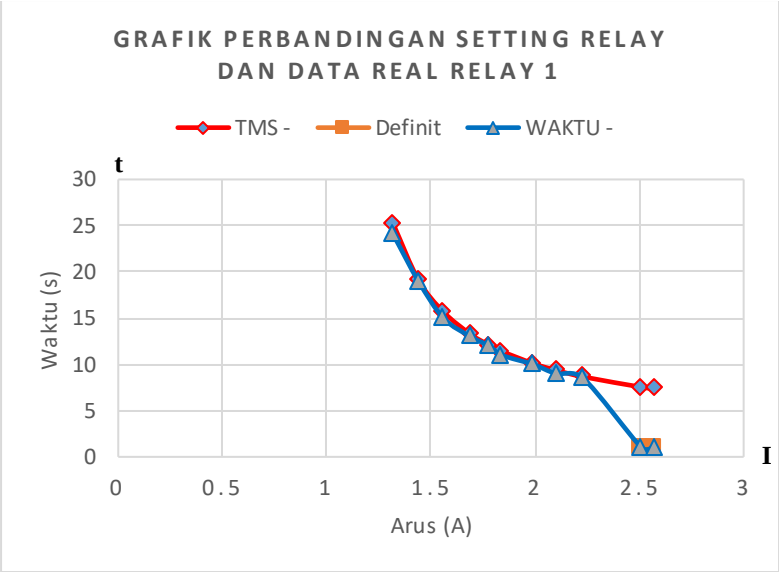
Setelah melakukan pengambilan data pada relai 2 maka langkah selanjutnya adalah melakukan pengambilan data pada relai 1 dengan nilai *setting* yang sama, pada Tabel 4.10 berikut ini merupakan hasil dari pengambilan data relai 1.

Tabel 4.10 Hasil Pengujian pada Relai 1

SET RELAI 2 [OL = (Iset=1A ; Timedial=1s), SC =(Isc=2.3A ; Timeset= 0.5s)]							
NO	Beban		I NOM	Setting		Data Nyata	
	Beban 1	Beban 2		TMS	Definite	Waktu	Kondisi
1	lampu 200 watt		0.88	-		-	normal

NO	Beban		I NOM	Setting		Data Nyata	
	Beban 1	Beban 2		TMS	Definite	Waktu	Kondisi
2	rice cooker (cook) 300 watt		1.32	25.14332061		24	Trip
3	rice cooker (cook) 300 watt	lampu 25 watt	1.44	19.1269374		19	Trip
4	rice cooker (cook) 300 watt	lampu 50 watt	1.56	15.67155461		15	Trip
5	rice cooker (cook) 300 watt	lampu 75 watt	1.69	13.27035384		13	Trip
6	rice cooker (cook) 300 watt	lampu 100 watt	1.78	12.06998498		12	trip
7	rice cooker (cook) 300 watt	lampu 125 watt	1.84	11.40996391		11	trip
8	rice cooker (cook) 300 watt	lampu 150 watt	1.99	10.10258847		10	trip
9	rice cooker (cook) 300 watt	lampu 175 watt	2.1	9.364932063		9	trip
10	rice cooker (cook) 300 watt	lampu 200 watt	2.23	8.658349429		8.5	trip
11	impedansi 440 watt	lampu 25 watt	2.51	7.536572643	1	1	Trip
12	impedansi 440 watt	lampu 50 watt	2.57	7.346213732	1	1	Trip

Berdasarkan Tabel 4.10 di atas beberapa beban yang di kombinasikan akan membuat urutan arus yang semakin tinggi, yang memiliki beberapa karakter dimana arus yang bernilai di bawah 1 A tidak akan membuat *trip* relai 1, sedangkan arus yang bernilai di atas 1 A akan membuat relai 1 dalam kondisi *trip*. waktu *trip* yang di peroleh dari nilai arus di atas 1 A dan di bawah 2.3 A memiliki waktu yang berbeda – beda. Semakin tinggi nilai arus pada nilai 1 sampai 2.3 A maka waktu *trip* akan semakin cepat, sedangkan waktu *trip* di atas 2.3 A memiliki waktu yang sama. Untuk membuktikan apakah nilai dari data arus dan waktu *trip* sudah sesuai dengan kurva *setting* arus *standard inverse* dan *definite*, maka di perlukan pembandingan data dengan menggunakan grafik yang merupakan hasil dari perbadingan data Relai 1.



Gambar 4.18 Grafik Perbandingan *Setting* Relai dan Data *Real* Relai 1

Berdasarkan Gambar 4.18 grafik di atas nilai waktu *trip* relai 1 antara 1 A sampai dengan 2.3 A tidak terlalu jauh dari nilai *setting*, sehingga data yang di dapat sesuai dengan karakteristik dari kurva invers, sedangkan waktu *trip* relai gangguan yang lebih dari 2.3 A memiliki waktu kerja yang sama dengan karakteristik kurva *definite*.

4.6 Pengambilan Data Koordinasi Relai

Dalam sistem proteksi, *digital relay* digunakan sebagai intrumen pengatur jalannya proteksi oleh karena itu koordinasi tiap relai sangat di perlukan agar sistem distribusi tenaga listrik lebih efisien saat terjadi gangguan Dalam proses proteksi terdapat dua jenis gangguan yaitu *over current* dan *short circuit*, pada prototipe sistem proteksi ini terdapat 2 buah relai di mana relai – relai tersebut saling terkoordinasi. Dalam proses koordinasi relai pengaturan nilai arus dan waktu sangatlah penting, oleh karena itu di butuhkan beberapa data koordinasi antar relai yaitu *over current test* dan *short circuit test*.

Over current test di lakukan dengan 2 cara yaitu yang pertama kondisi dimana *time dial* pada kedua relai sama, namun *setting* arus *over current* berbeda, sedangkan yang kedua lakukan dengan cara mengatur *time dial* pada kedua relai yang berbeda namun arus *over current* sama. Hasil dari tes *over current* di tampilkan dalam bentuk Tabel seperti berikut.

Tabel 4.11 Data Pengujian Koordinasi *Over Current Test*

over current test														
no	relay 1		relay 2		beban		I nom		waktu trip		waktu hitung		kondisi	
	Iset	timedial	Iset	timedial	busbar 1	busbar 2	relay 1	relay 2	relay 1	relay 2	relay 1	relay 2	relay 1	relay 2
1	1	1.2	1	0.6	-	1.35	1.33	1.35	30	14	29.4	14	-	TRIP
2	1	1.2	1	0.6	1.34	-	1.34	-	29	-	28.6	-	TRIP	-
3	1	1.2	1	0.6	0.74	1.36	2.07	1.36	10	12	11.5	13.6	TRIP	-
4	1	1.2	1	0.6	1.32	0.75	2.07	0.75	10	-	11.5	-	TRIP	-
5	1.2	0.6	1	0.6	-	1.36	1.34	1.36	38	13.5	38	13.6		TRIP
6	1.2	0.6	1	0.6	1.35	-	1.35	-	36	-	35.6	-	TRIP	-
7	1.2	0.6	1	0.6	0.72	1.35	2.06	1.35	8	14	7.73	14	TRIP	-
8	1.2	0.6	1	0.6	1.31	0.75	2.06	0.75	8	-	7.73	0	TRIP	-

Tes pengujian berikutnya yaitu *short circuit test*. *Short circuit test* di lakukan dengan 2 cara yaitu yang pertama kondisi dimana *time trip* kedua relai sama namun *setting* arus *short circuit* berbeda, sedangkan yang

kedua di lakukan dengan cara mengatur *time trip* kedua relai berbeda namun arus *short circuit* sama. Hasil dari *Short circuit test* di tampilkan dalam bentuk Tabel seperti berikut.

Tabel 4.12 Data Pengujian Koordinasi *Short Circuit Test*

short circuit test												
no	relay 1		relay 2		beban		I nom		waktu trip		kondisi	
	Iset	time set	I set	time set	busbar 1	busbar 2	busbar 1	busbar 2	relay 1	relay 2	relay 1	relay 2
1	2.1	2	2.1	1	-	short (2.21A)	2.21	2.21		1		trip
2	2.1	2	2.1	1	short (2.22A)	0.77	0.77	2.97	2		trip	
3	2.5	1	2.3	1	0.26	short (2.21A)	2.48	2.22		1		trip
4	2.5	1	2.3	1	short (2.22A)	0.39	2.57	0.39	1		trip	

Keterangan:

Beban busbar 1 = beban yang ada pada busbar 1 (Ampere)
Beban busbar 2 = beban yang ada pada busbar 2 (Ampere)
I nom relay = arus yang mengalir pada relay 1(Ampere)
I nom relay = arus yang mengalir pada relay 1(Ampere)
Waktu trip relay 1 = pembacaan waktu yang di butuhkan relay 1 untuk trip dengan mengunakan stopwatch (detik)
Waktu trip relay 2 = pembacaan waktu yang di butuhkan relay 2 untuk trip dengan mengunakan stopwatch (detik)
Waktu hitung relay 1 = perhitungan waktu yang di butuhkan relay 1 untuk trip berdasarkan perhitungan kurva proteksi inverse (detik)
Waktu hitung relay 2 = perhitungan waktu yang di butuhkan relay 2 untuk trip berdasarkan perhitungan kurva proteksi inverse (detik)
Kondisi relay 1 = kondisi relay 1 setelah ada gangguan pada jaringan
Kondisi relay 1 = kondisi relay 1 setelah ada gangguan pada jaringan

4.7 Analisa Relevansi

Pada alat Tugas Akhir kami ini merupakan perancangan untuk *monitoring* dan juga *setting* dengan kendali jarak jauh pada sistem proteksi kelistrikan. Untuk proteksi pada sistem kelistrikan ini, kami membuat 2 relai proteksi sebagai simulasi koordinasi relai yang bisa diterapkan pada keadaan sebenarnya. Untuk *monitoring* dan *setting* ini kami menggunakan *software* Lazarus dengan media komunikasi RS485 yang bisa menyambungkan maksimal 32 *master* dengan 32 *slave*. Apabila akan diimplementasikan pada kehidupan sebenarnya, maka perlu ada penyesuaian spesifikasi komponen yang digunakan pada relai pada keadaan sebenarnya, karena pada alat ini kemampuan pengukuran hanya dibatasi pada beban dengan batas arus maksimal kurang lebih 3 A dan hanya dapat digunakan untuk jaringan 1 fasa (karena hanya smulasi). Untuk melakukan proteksi pada saluran 3 fasa seperti pada gardu induk, maka perlu dilakukan penambahan dan penggantian komponen yang sesuai dengan keadaan sebenarnya pada alat Tugas Akhir kami ini.

Proses *monitoring* dan juga *setting* jarak jauh ini sudah dapat dilakukan asalkan hardware sudah dihubungkan dengan benar pada komputer agar terdeteksi pada *interface* nya yaitu menggunakan *software* Lazarus. Diharapkan dengan adanya alat *monitoring* dan juga *setting* jarak jauh ini maka dapat membantu pengguna alat ini (*user*) terutama pada perusahaan listrik negara Indonesia ini yaitu PT. PLN (Persero), dalam memantau kondisi sistem kelistrikan di Indonesia dan juga dapat memberikan *setting* sesuai keperluan dengan cara yang lebih mudah. Diharapkan juga dapat membantu PLN untuk mencegah adanya kerusakan pada peralatan konsumen dikarenakan adanya kesalahan koordinasi yang terjadi pada sistem proteksi.

BAB V
PENUTUP

Bab penutup ini berisi kesimpulan yang diperoleh selama proses pembuatan tugas akhir yang berjudul **"Monitoring Dan Setting Digital Relay Satu Fasa Terhadap Gangguan Over Current"**, serta saran untuk pengembangan alat ini kedepannya.

5.1 Kesimpulan

Berdasarkan hasil pengujian pada Tugas Akhir ini, maka diperoleh beberapa kesimpulan yaitu :

1. Hal pertama kali yang diperlukan dalam pembuatan alat ini adalah kalibrasi sensor arus. Pada alat kami ini pada pengukuran awal terdapst *error* sekitar 4,73 % lalu saat dilakukan kalibrasi dengan alat yang sudah terkalibrasi yaitu Fluke 5500 didapatkan *error* sekitar 0.62 %.
2. Kondisi arus dan tegangan pada alat kami akan disimpan pada SD *card* beserta waktu terkini yang terbaca pada kedua relai dengan menggunakan RTC.
3. Komunikasi antara komputer dengan kedua relai bisa mencapai jarak 1200 m dikarenakan menggunakan media RS485 dan dapat ditampilkan pada *interface software* Lazarus dikarenakan menggunakan protokol Modbus.

5.2 Saran

Beberapa saran yang bisa kami usulkan untuk mengembangkan tugas akhir ini yaitu:

1. Digunakannya genset sebagai sumber tenaga pada simulasi alat kami ini, diharapkan dengan menggunakan genset dapat mendekati kondisi nyata dan juga dapat memperbesar batasan arus untuk penelitian yang lebih presisi.
2. Diharapkan pada alat ini bisa dioperasikan pada jaringan 3 fasa agar kondisinya lebih nyata seperti proteksi relai pada jaringn tegangan menengah maupun pada gardu induk.\

-----Halaman ini sengaja dikosongkan-----

DAFTAR PUSTAKA

[1] ...., **PENGESAHAN RUPTL PLN TAHUN 2016-2025**, MENTERI ENERGI DAN SUMBER DAYA MINERAL REPUBLIK INDONESIA, 2016

[2] Andrianto, Heri; dan Aan Darmawan. 2016. **Arduino Belajar Cepat dan Pemrograman**. Bandung : INFORMATIKA.

[3] ...., **IDE *Software for* Arduino**, Datasheet, 2015.

[4] ...., **INFORCE UNINTERRUPTIBLE POWER SUPPLY 650 VA**, Manual Book, 2009.

[5] DFROBOT, **ARDUINO LCD KEYPAD DFROBOT**, https://www.dfrobot.com/, 19 Mei 2017.

[6] …., **OPEN ENERGY MONITOR,** https://openenergymonitor.org/, 15 April 2017

[7] **AZZETTLER *RELAYS***, http://www.azettler.com, 16 April 2017

[8] Adhitya, W.W. dan Faisal, A., **PERANCANGAN SISTEM MONITORING VOLTAGE FLICKER BERBASIS ARDUINO DENGAN METODE FAST FOURIER TRANSFORM (FFT)**, Tugas Akhir, Program D3 Teknik Elektro FTI-ITS, Surabaya, 2016.

[9] …., **BUKU PANDUAN PUSDIKLAT PLN BAB VII**, RELAI ARUS LEBIH**,** 2016

[10] …., **BUKU PANDUAN PUSDIKLAT PLN BAB IX**, KOORDINASI PROTEKSI DISTRIBUSI, 2016

-----Halaman ini sengaja dikosongkan-----

LAMPIRAN A

A.1 Listing Program pada Arduino

```
/*ReleComb.h*/
// Pin Assigment
#define PIN_TRIP_SWITCH 39 // Test Switch
#define PIN_RESET_SWITCH 41 // Reset Switch
#define PIN_RELAY_OUT 11 // Relay Output
#define PIN_LED_LIVE 13 // LED live
#define PIN_TX_RX_ENABLE 3 // RS-485 enable pin
#define SLAVE_ADDR_PIN0 35 // Slave ID Address0
#define SLAVE_ID 2 // Address Modbus Slave
#define ALAMAT_EEPROM_AWAL_EVENT 101 // Address
Awal Untuk Data Logger Gangguan
#define ALAMAT_EEPROM_AWAL_DATA 0 // Address
Awal Untuk Data Relay
#define JUMLAH_EVENT_MAX 50 // Jumlah Maksimal Data
Logger Gangguan
#define CROSSING_MAX 2 // EmonLib Crossing 20 kali
#define TIME_OUT_MAX 50 // EmonLib Time Out 2000 ms
#define SHORT_CIRCUIT 1 //Status Short Circuit
#define OVER_LOAD 2 //Status Over Load
// Operational from Host
#define NO_OPERATION 0
#define DEFAULT_SET 1
#define CHANGE_SET 2
#define READ_SET 3
#define CLEAR_EVENT 4
#define READ_EVENT 5
#define TRIP_REMOTE 6
#define RESET_REMOTE 7
#define READ_HISTORY_SET 8

// LCD pin setting
// For LCD Keypad DF Robot
#define LCD_RS 8
#define LCD_EN 9
#define LCD_D4 4
```

```
#define LCD_D5  5
#define LCD_D6  6
#define LCD_D7  7
#define PIN_KEYPAD  A0 // pin Analog Keypad
#define LCD_ROWS  2
#define LCD_COLUMNS  16
//
#define BUTTONS_PER_COLUMN  5 // Each analog pin has five
buttons with resistors.
#define BUTTONS_PER_ROW  1 // There are two analog pins in
use.

#define  REMOTE_RESET_ACTIVE  1//Reset dari Host
#define  REMOTE_TRIP_ACTIVE  1//Trip dari Host

#define  MAX_DETIK  60
#define  BAUD_RATE  9600  //untuk serial comm
#define  A20MILLISECOND  20 //20 ms
#define  SEQ_CYCLE_SPEED  20 //20 ms
#define  WORD_MAX  0xFFFF//maksimum  16 bit
#define  BYTE_MAX  0xFF//maksimum  8 bit
#define  MAX_PV  10.0  // Nilai  Arus Maksimum
#define  MAX_TEG  500.0  // Nilai  Teg Maksimum

// For Menu Index
#define MAIN_MENU_IDX  0 //Index Main Menu
//-----------------
#define TRIP_RESET_MENU_IDX  10 //Index Short Cicuit  Menu
#define TRIP_MENU_IDX  1 //Index KeyPad Trip
#define RESET_MENU_IDX  2 //Index Short Cicuit  Reset
#define BACK_MENU_IDX  3 //Index Short Cicuit  Reset
//-----------------
#define DISPLAY_MENU_IDX  20 //Index Display Menu
//-----------------
#define STATUS_MENU_IDX  30 //Index Display Menu
//-----------------
#define INFO_MENU_IDX  40 //Index Display Menu

struct IdMenu
```

```
{
  byte mainMenu;
  byte subMenu;
};

struct DataEvent {
   byte   NomorEvent;
   float  Arus;
   float  Tegangan;
   byte   Status;
   byte   EventType;
   byte   Detik;
   byte   Menit;
   byte   Jam;
   byte   Tanggal;
   byte   Bulan;
   int    Tahun;
};

struct DataSettingRelay {
   byte   TChar;
   byte   Action;
   float  SetPoint;
   unsigned int  TMS;
   unsigned int  TSet;

   byte   Version;
   byte   Detik;
   byte   Menit;
   byte   Jam;
   byte   Tanggal;
   byte   Bulan;
   int    Tahun;
};
DataSettingRelay DataReleShortCircuit,DataReleOverLoad;

enum
{
  // just add or remove registers and your good to go...
```

```
  // The first register starts at address 0
//Data from Ardu
  ARDU_ID, //index 0
//For Short Circuit
  ARDU_SH_SP, //index 1
  ARDU_SH_PV, //index 2
  ARDU_SH_ACTION, //index 3
  ARDU_SH_TCHAR, //index 4
  ARDU_SH_TSET, //index 5
  ARDU_SH_TMS, //index 6
  ARDU_SH_TACC, //index 7
  ARDU_SH_STATE, //index 8
//For Over Load
  ARDU_OL_SP, //index 9
  ARDU_OL_PV, //index 10
  ARDU_OL_ACTION, //index 11
  ARDU_OL_TCHAR, //index 12
  ARDU_OL_TSET, //index 13
  ARDU_OL_TMS, //index 14
  ARDU_OL_TACC, //index 15
  ARDU_OL_STATE, //index 16
  ARDU_EVENT_NBR, //index 17
//Operational from Host
  HOST_OPR_MODE, //index 18
  HOST_OPR_STATUS, //index 19
  LAST_OPR_MODE, //index 20
//for setting data Short Circuit
  HOST_SH_SP, //index 21
  HOST_SH_ACTION, //index 22
  HOST_SH_TCHAR, //index 23
  HOST_SH_TSET, //index 24
  HOST_SH_TMS, //index 25
//for setting data Over Load
  HOST_OL_SP, //index 26
  HOST_OL_ACTION, //index 27
  HOST_OL_TCHAR, //index 28
  HOST_OL_TSET, //index 29
  HOST_OL_TMS, //index 30
//tgl 13/05/2017 --- for setting date
```

```
  HOST_SET_VERSION,  //index 31
  HOST_SET_TAHUN,  //index 32
  HOST_SET_BULAN,  //index 33
  HOST_SET_TGL,  //index 34
  HOST_SET_JAM,  //index 35
  HOST_SET_MENIT,  //index 36
  HOST_SET_DETIK,  //index 37
//tgl 13/05/2017  --- for event data
  HOST_EVENT_WILL_READ,  //index 38, nomor event yang
akan dibaca
  HOST_EVENT_NUMBER,  //index 39
  HOST_EVENT_TAHUN,  //index 40
  HOST_EVENT_BULAN,  //index 41
  HOST_EVENT_TGL,  //index 42
  HOST_EVENT_JAM,  //index 43
  HOST_EVENT_MENIT,  //index 44
  HOST_EVENT_DETIK,  //index 45
  HOST_EVENT_STATUS,  //index 46
  HOST_EVENT_ARUS,  //index 47
  HOST_EVENT_TEG,  //index 48
  HOST_EVENT_TYPE,  //index 49
  HOST_DIAG1,  //index 50
  HOST_DIAG2,  //index 51
// leave this one
  HOLDING_REGS_SIZE
  // total number of registers for function 3 and 16 share the same
register array
};
unsigned int holdingRegs[HOLDING_REGS_SIZE];  // function 3
and 16 register array

/*RelayArdu*/
/*
 *  Program ini dirancang untuk Relay Proteksi
 *  - Version 0.0, 8/5/2017
*/
#if ARDUINO  < 100
#include <WProgram.h>
#else
```

```
#include <Arduino.h>
#endif

#include "EmonLib.h"  //Library sensor arus dan tegangan
#include "ReleComb.h"  //Untuk inisialisai pin dll
#include <RTClib.h>  //Library RTC
#include <Wire.h> //Untuk 12C
#include <RelayProt.h> //Library relay proteksi
#include <SequenceTimer.h>//Library  waktu millis

#include <SD.h> //Library SDCard
#include <EEPROM.h>  //Library EEPROM
#include <LiquidCrystal.h>  // Library LCD
#include <phi_interfaces.h> //Library keypad
#include <phi_prompt.h> //Library LCD display
#include <SimpleModbusSlave.h>  //Library protokol Modbus
Slave

volatile int          itrNmbr;  //niai interrupt
volatile boolean sudahTulisEvent; //pernyataan sudah menulis ke
EEPROM

EnergyMonitor monitoringArusDanTegangan;  //Create an
instance for Emonlib

DataEvent DataTulis; //Buat menulis ke EEPROM
RTC_DS1307  rtc; //Deklarasi pake RTC DS1307
//RTC_Millis  rtc; //Deklarasi buat ngambil waktu dari laptop,
simulasi
DateTime  SaatIni; //Waktu saat ini

unsigned long prevMilli;  //Millis sebelumnya
byte keypad_type=Analog_keypad; //Memilih tipe keypad analog
char mapping[]={'R','U','D','L','S'};  //Right, Up, Down, Left, Select
byte pins[]={PIN_KEYPAD};  //The pin numbers are analog pin
numbers
int values[]={5, 105, 260, 415, 640}; //value for DF Robot LCD
Keypad
```

```
phi_analog_keypads analogKeypad(mapping, pins, values,
BUTTONS_PER_ROW, BUTTONS_PER_COLUMN);
//Mapping analogkeypad
multiple_button_input * keypads[]={&analogKeypad,0};
char up_keys[]={"U"}; ///< All keys that act as the Up key are
listed here.
char down_keys[]={"D"}; ///< All keys that act as the Down key
are listed here.
char left_keys[]={"L"}; ///< All keys that act as the Left key are
listed here.
char right_keys[]={"R"}; ///< All keys that act as the Right key are
listed here.
char enter_keys[]={"S"}; ///< All keys that act as the Select key are
listed here.
char *
function_keys[]={up_keys,down_keys,left_keys,right_keys,enter_
keys}; ///< All function key names are gathered here fhr
phi_prompt.
// LCD
LiquidCrystal
lcd(LCD_RS,LCD_EN,LCD_D4,LCD_D5,LCD_D6,LCD_D7);
//deklarasi pin LCD
int global_style=109; // This is the style of the menu
// SD Card
File myFile;
int pinCS = 53;
// Relay Proteksi
RelayProt RelayShortCirt("RlShortCirt"); //Deklarasi relay Short
Circuit
RelayProt RelayOverLoad("RlOverLoad"); //Deklarasi relay Over
Load
SequenceTimer SequenceUtama(SEQ_CYCLE_SPEED); //dalam
mili detik

IdMenu menuId;

void setup() {
  // put your setup code here, to run once:
```

```
  // Timer0 is already used for millis() - we'll just interrupt
somewhere
  // in the middle and call the "Compare A" function below
  OCR0A = 0xAF;
  TIMSK0 |= _BV(OCIE0A);
  prevMilli = millis();

  setupModbus (); //Setup Modbus, Addressing dan konfigurasi

  arduInterfaceSetup(); //Setup Keypad-LCD
  pinMode(PIN_LED_LIVE, OUTPUT); //LED pin 13

  SetupEmonlib(); //Setup baca arus dan tegangan

//Setup Relay
  setupRelayShortCirt(); //setup relay protection untuk short circuit
  setupRelayOverLoad(); //setup relay protection untuk overLoad,
25/05/2017
  setupDefault(); //Setting Default

  updateDataRelayFromEEPROM(); //Ambil Data Dari EEPROM
Dan Validasi, saat pertama kali jalan
//.............
  readEventFromEEPROM(); //Ambil Event Dari EEPROM, jika
valid transfer to HoldingRegs

  transferFromProcessToHoldingRegsArdu(); //Transfer dari Relay
parameter ke HoldingRegsArdu
  equelizingHostData(); //Menyamakan awal HoldingRegsHost dan
HoldingRegsArdu

  SequenceUtama.setSetingMiliSecond(500); //Setup Sequence
utama 500ms

//semua terkait dengan menu
  setupMenuReleComb(); //lihat file MenuReleComb
  subMenuInfo(); //Tampilkan display info Relecomb ke LCD

//terkait RTC dan SD Card
```

```
  // Setup SD Card
  SD.begin();
  ///*
  //RTC DS1307
  rtc.begin();
  //*/
  /*
  //Software RTC
    rtc.begin(DateTime(F(__DATE__),  F(__TIME__)));
  */
}

void loop() {
  // put your main code here, to run repeatedly:
  // Running on 20 mili Second
  if (SequenceUtama.isMiliSecondEvent()) { //Dijalankan tiap 500
ms
    updateCommandFromHost(); //Check data dari Host dulu dan
transfer sebagian ke operasi
    transferFromProcessToHoldingRegsArdu(); //Semua Proses
disimpan ke HoldingRegs
    mainMenuRele(); //Menampilkan menu LCD
  }
  if (SequenceUtama.isASecondEvent()){ //Dijalankan setiap detik

monitoringArusDanTegangan.calcVI(CROSSING_MAX,TIME_O
UT_MAX); // Calculate all. No.of half wavelengths (crossings),
time-out
    digitalWrite( PIN_LED_LIVE, digitalRead(PIN_LED_LIVE) ^
1 ); //Jalankan LED pin 13
    SaatIni = rtc.now(); //Deklarasi waktu saat ini
    DataLog(); //Simpan datalogger ke SD Card
  }
  modbus_update(); //Deklarasi protokol Modbus
}

//Lokasi penempatan prosedur
void arduInterfaceSetup(){
  lcd.begin(LCD_COLUMNS,  LCD_ROWS);
```

```
  init_phi_prompt(&lcd,keypads,function_keys, LCD_COLUMNS,
LCD_ROWS,  '~'); // Supply the liquid crystal object, input
keypads, and function key names. Also supply the column and row
of the lcd, and indicator as '>'. You can also use '\x7e', which is a
right arrow.
}

void setupDefault(){
  /*setup Relay Protection Short Circuit*/
  RelayShortCirt.setTimeChar(DEF_TIME);//Time  Characteristic
  RelayShortCirt.setTimeDef(500);
  RelayShortCirt.setSetting(3.0);//set  setting value
  /*setup Relay Protection Over Load (Tambahan 20/05/2017)*/
  RelayOverLoad.setTimeChar(INVS_STD);//Time  Characteristic
  RelayOverLoad.setTimeDef(1500);
  RelayOverLoad.setSetting(1.5);//set  setting value
}

void setupRelayShortCirt(){
  //setup Relay Protection Short Circuit
  RelayShortCirt.setPinTest(PIN_TRIP_SWITCH);
  RelayShortCirt.setPinReset(PIN_RESET_SWITCH);
  RelayShortCirt.setPinRL(PIN_RELAY_OUT);
  RelayShortCirt.setActionChar(OVR_ACTION);
  RelayShortCirt.setActive(true);
}

void setupRelayOverLoad(){
  //setup Relay Protection Over Load
  RelayOverLoad.setPinTest(PIN_TRIP_SWITCH);
  RelayOverLoad.setPinReset(PIN_RESET_SWITCH);
  RelayOverLoad.setPinRL(PIN_RELAY_OUT);
  RelayOverLoad.setActionChar(OVR_ACTION);
  RelayOverLoad.setActive(true);
}

 void equelizingHostData(){
/*
//For Short Circuit
```

```
  ARDU_SH_SP,  //index 1
  ARDU_SH_PV,  //index 2
  ARDU_SH_ACTION,  //index 3
  ARDU_SH_TCHAR,  //index 4
  ARDU_SH_TSET,  //index 5
  ARDU_SH_TMS,  //index 6
  ARDU_SH_TACC,  //index 7
 //For Over Load
  ARDU_OL_SP,  //index 8
  ARDU_OL_PV,  //index 9
  ARDU_OL_ACTION,  //index 10
  ARDU_OL_TCHAR,  //index 11
  ARDU_OL_TSET,  //index 12
  ARDU_OL_TMS,  //index 13
  ARDU_OL_TACC,  //index 14
 //for setting data Short Circuit
  HOST_SH_SP,  //index 20
  HOST_SH_ACTION,  //index 21
  HOST_SH_TCHAR,  //index 22
  HOST_SH_TSET,  //index 23
  HOST_SH_TMS,  //index 24
 //for setting data Over Load
  HOST_OL_SP,  //index 25
  HOST_OL_ACTION,  //index 26
  HOST_OL_TCHAR,  //index 27
  HOST_OL_TSET,  //index 28
  HOST_OL_TMS,  //index 29
 */
 //Short Circuit = Host --> Ardu
  holdingRegs[HOST_SH_SP] = holdingRegs[ARDU_SH_SP];
  holdingRegs[HOST_SH_ACTION] =
holdingRegs[ARDU_SH_ACTION];
  holdingRegs[HOST_SH_TCHAR] =
holdingRegs[ARDU_SH_TCHAR];
  holdingRegs[HOST_SH_TSET] =
holdingRegs[ARDU_SH_TSET];
  holdingRegs[HOST_SH_TMS] =
holdingRegs[ARDU_SH_TMS];
 //Over Load = Host --> Ardu
```

```
  holdingRegs[HOST_OL_SP]  = holdingRegs[ARDU_OL_SP];
  holdingRegs[HOST_OL_ACTION]  =
holdingRegs[ARDU_OL_ACTION];
  holdingRegs[HOST_OL_TCHAR]  =
holdingRegs[ARDU_OL_TCHAR];
  holdingRegs[HOST_OL_TSET]  =
holdingRegs[ARDU_OL_TSET];
  holdingRegs[HOST_OL_TMS]  =
holdingRegs[ARDU_OL_TMS];
}

// Interrupt is called once a millisecond,
SIGNAL(TIMER0_COMPA_vect)
{
  unsigned long currMilli,  delta;
  currMilli  = millis();
  delta = currMilli  - prevMilli;
  SequenceUtama.execute();
  itrNmbr++;
  if (delta >= A20MILLISECOND){
  //Jalankan Fungsi Proteksi
  RelayShortCirt.execute(monitoringArusDanTegangan.Irms,
millis());  //Beri nilai  dan jalankan  (Short Circuit)
  RelayOverLoad.execute(monitoringArusDanTegangan.Irms,
millis());  //Beri nilai  dan jalankan  (Over Load)
  if (RelayShortCirt.getState()  >= STATUS_TRIP)  {
    //Tulis  Event Ke EEPROM
    MenulisEventKeEEPROM(SHORT_CIRCUIT);  //Simpan data
gangguan ke EEPROM
    sudahTulisEvent = false;
  }
  else if (RelayOverLoad.getState()  >= STATUS_TRIP)  {
    //Tulis  Event Ke EEPROM
    MenulisEventKeEEPROM(OVER_LOAD);//  Simpan data
gangguan ke EEPROM
    sudahTulisEvent = false;
  }
  else sudahTulisEvent = true;
    itrNmbr=0;
```

```
    prevMilli = currMilli;
  }
}

void SetupEmonlib()
{
  monitoringArusDanTegangan.voltage(1, 233, 1.7);  // Voltage:
input pin, calibration, phase_shift (RL1)
  //monitoringArusDanTegangan.voltage(1, 250, 1.7);  // Voltage:
input pin, calibration, phase_shift (RL2)
  //monitoringArusDanTegangan.current(3, 10);  // Current: input
pin, calibration. (default)
  monitoringArusDanTegangan.current(3,
9.68081395348837209302325581395 35);  // Current: input pin,
calibration. (RL1)
  //monitoringArusDanTegangan.current(3,
10.480239018765069713806478666527);  // Current: input pin,
calibration. (RL2)
}

void DataLog(){
char Time[20];  //Pendeklarasian Waktu
char Date[20];  //Pendeklarasian Tanggal
  sprintf(Time, "%02d:%02d:%02d",  SaatIni.hour(),
SaatIni.minute(), SaatIni.second());
  sprintf(Date, "%02d/%02d/%02d", SaatIni.day(), SaatIni.month(),
SaatIni.year());
  myFile = SD.open("MMC.txt", FILE_WRITE);  //membuka file
dan menulis isi filenya
  if (myFile) {
    myFile.print(Date);  // print ke file yang ada di micro SD
    myFile.print(",");
    myFile.print(Time);
    myFile.print(",");
    myFile.print(monitoringArusDanTegangan.Irms);
    myFile.print(",");
    myFile.println(monitoringArusDanTegangan.Vrms);
    myFile.close();  // tutup file
  }
```

```
  // jika file tidak dapat dibuka, print error.
  else {
    Serial.println("error opening test.txt");
  }
}

void MenulisEventKeEEPROM(byte EventVal){
  if (sudahTulisEvent == true){
    byte JumlahEvent =
EEPROM.read(ALAMAT_EEPROM_AWAL_EVENT-1);
//Mengetahui jumlah event terakhir
    if (JumlahEvent == 0xFF)JumlahEvent = 1; //Antisipasi jika
Arduino belum pernah menulis event
    JumlahEvent++;
    if (JumlahEvent >= JUMLAH_EVENT_MAX) JumlahEvent =
1;
    int AlamatEvent = ALAMAT_EEPROM_AWAL_EVENT +
(JumlahEvent-1)*sizeof(DataEvent);
    MengisiDataEvent(JumlahEvent, EventVal);
    EEPROM.put(AlamatEvent, DataTulis);
    EEPROM.write(ALAMAT_EEPROM_AWAL_EVENT-
1,JumlahEvent);
    transferEventToHoldingRegs(DataTulis); //Update HoldingRegs
  }
}

void readEventFromEEPROM(){
  byte tempEventNbr =
EEPROM.read(ALAMAT_EEPROM_AWAL_EVENT-1);
  if ((tempEventNbr > 0) && (tempEventNbr <=
JUMLAH_EVENT_MAX)){

transferEventToHoldingRegs(MembacaEventDariEEPROM(temp
EventNbr));
  }
}
DataEvent MembacaEventDariEEPROM(byte NomorEvent){
  DataEvent tempEvent;
```

```
  int AlamatEvent = ALAMAT_EEPROM_AWAL_EVENT +
(NomorEvent-1)*sizeof(DataEvent);
  EEPROM.get(AlamatEvent, tempEvent);
  return tempEvent;
}

void MengisiDataEvent(byte NomorEvent, byte EventVal){
  /*Status Short Circuit dan Over Load*/
  if (EventVal == SHORT_CIRCUIT){
    DataTulis.Arus = RelayShortCirt.getValue();
    DataTulis.Status = RelayShortCirt.getState();
  }
  if (EventVal == OVER_LOAD){
    DataTulis.Arus = RelayOverLoad.getValue();
    DataTulis.Status = RelayOverLoad.getState();
  }
  DataTulis.EventType = EventVal;
  DataTulis.Tegangan = monitoringArusDanTegangan.Vrms;
  DataTulis.NomorEvent = NomorEvent;
  DataTulis.Detik =SaatIni.second() ;
  DataTulis.Menit = SaatIni.minute();
  DataTulis.Jam = SaatIni.hour();
  DataTulis.Tanggal = SaatIni.day();
  DataTulis.Bulan = SaatIni.month();
  DataTulis.Tahun = SaatIni.year();
}

//Untuk Menentukan Alamat Komunikasi
byte getSlaveId(){
  byte _slaveID=0;
  if (digitalRead(SLAVE_ADDR_PIN0))_slaveID=_slaveID |
B00000001;
  return (_slaveID+1);
}
//Setup Modbus, Addressing dan konfigurasi
void setupModbus(){
  pinMode(SLAVE_ADDR_PIN0, INPUT);//untuk alamat Slave

  byte slaveId = getSlaveId();
```

```
  modbus_configure(&Serial1, BAUD_RATE, SERIAL_8N2,
slaveId, PIN_TX_RX_ENABLE, HOLDING_REGS_SIZE,
holdingRegs);
}

/*HostComm*/
void transferFromProcessToHoldingRegsArdu(){
  holdingRegs[ARDU_ID] = getSlaveId();
//Proses --> HoldingRegs (Short Circuit)
  holdingRegs[ARDU_SH_PV] =
RelayShortCirt.getValue()*WORD_MAX/MAX_PV;
  holdingRegs[ARDU_SH_STATE] = RelayShortCirt.getState();
  holdingRegs[ARDU_SH_SP] =
RelayShortCirt.getSetting()*WORD_MAX/MAX_PV;
  holdingRegs[ARDU_SH_TCHAR] =
RelayShortCirt.getTimeChar();
  holdingRegs[ARDU_SH_TSET] = RelayShortCirt.getTimeDef();
  holdingRegs[ARDU_SH_TMS] = RelayShortCirt.getTMS();
  holdingRegs[ARDU_SH_TACC] =
RelayShortCirt.getTimeACC();
  holdingRegs[ARDU_SH_ACTION] =
RelayShortCirt.getActionChar();
//Proses --> HoldingRegs (Over Load)
  holdingRegs[ARDU_OL_PV] =
RelayOverLoad.getValue()*WORD_MAX/MAX_PV;
  holdingRegs[ARDU_OL_STATE] = RelayOverLoad.getState();
  holdingRegs[ARDU_OL_SP] =
RelayOverLoad.getSetting()*WORD_MAX/MAX_PV;
  holdingRegs[ARDU_OL_TCHAR] =
RelayOverLoad.getTimeChar();
  holdingRegs[ARDU_OL_TSET] =
RelayOverLoad.getTimeDef();
  holdingRegs[ARDU_OL_TMS] = RelayOverLoad.getTMS();
  holdingRegs[ARDU_OL_TACC] =
RelayOverLoad.getTimeACC();
  holdingRegs[ARDU_OL_ACTION] =
RelayOverLoad.getActionChar();
  //Untuk jumlah event
```

```
  holdingRegs[ARDU_EVENT_NBR]  =
EEPROM.read(ALAMAT_EEPROM_AWAL_EVENT-1);
//Mengetahui jumlah event terakhir
}

void updateCommandFromHost(){
  DataEvent tempEvent;
  word startAddr, lenghtAddr;
  byte OprStatus,LastOprStatus;
  byte HostCommand = holdingRegs[HOST_OPR_MODE];

  byte valStatus = RelayShortCirt.getState();
  if (valStatus < RelayOverLoad.getState())valStatus =
RelayOverLoad.getState(); //Ambil yang tertinggi

//Operational EEPROM
  OprStatus = 0;
  LastOprStatus = OprStatus;
  switch (HostCommand)
  {
    case TRIP_REMOTE:
      OprStatus++;
      RelayShortCirt.setState(STATUS_TEST_REMOTE);
      OprStatus++;
      RelayOverLoad.setState(STATUS_TEST_REMOTE);
      OprStatus++;
    break;
    case RESET_REMOTE:
      //Reset tidak bisa dioperasikan jika LocalTrip aktif
      OprStatus++;
      valStatus = valStatus & STATUS_TEST_LOCAL;  //Ambil bit
test lokal
      if (valStatus != STATUS_TEST_LOCAL){
        OprStatus++;
        RelayShortCirt.setReset(true);
        OprStatus++;
        RelayOverLoad.setReset(true);
        OprStatus++;
      }
```

```
    break;
    case CHANGE_SET://
      OprStatus++;
      if (holdingRegs[HOST_SET_VERSION]  >= 0){ //Check dan
jalankan  jika  ada update Setting dari Host/Master
        OprStatus++;
        if (DataReleShortCircuit.Version  !=
holdingRegs[HOST_SET_VERSION]){  //Check dan jalankan jika
ada update Setting dari Host/Master
          OprStatus++;
          OprStatus = OprStatus +
transferFromHostHoldingRegsToDataRelay();  //transfer to
dataRelay, process and write  to EEPROM
        }
      }
    break;
    case DEFAULT_SET:  //Kembali  ke Setting  Default
      OprStatus++;
      if (holdingRegs[HOST_SET_VERSION]  > 0 ){
        startAddr = ALAMAT_EEPROM_AWAL_DATA;
        lenghtAddr = 2 * sizeof(DataSettingRelay);  //Setting
ShortCircuit  dan OverCurrent
        OprStatus++;
        ClearEEPROM(startAddr,lenghtAddr);
        OprStatus++;
        setupDefault(); //Setting Default
        OprStatus++;
        transferFromProcessToHoldingRegsArdu(); //Transfer dari
Relay parameter  ke HoldingRegsArdu
        OprStatus++;
        equelizingHostData();  //Menyamakan awal HoldingRegsHost
dan HoldingRegsArdu
        holdingRegs[HOST_SET_VERSION]  = 0;
        OprStatus++;
      }
    break;
    case READ_EVENT:  //Baca Event
      OprStatus++;
      if (holdingRegs[HOST_EVENT_WILL_READ]  != 0 ){
```

```
    OprStatus++;
     tempEvent =
MembacaEventDariEEPROM(holdingRegs[HOST_EVENT_WIL
L_READ]);
     OprStatus++;
     if (holdingRegs[HOST_EVENT_WILL_READ] ==
tempEvent.NomorEvent){  //Valid event number
      OprStatus++;
      transferEventToHoldingRegs(tempEvent);
      OprStatus++;
     }
    }
    //Akhir dari baca All Event
    if ((holdingRegs[HOST_EVENT_WILL_READ] == 0 ) &&
(holdingRegs[HOST_EVENT_NUMBER] > 0 )){
     OprStatus++;
      tempEvent =
MembacaEventDariEEPROM(holdingRegs[HOST_EVENT_NUM
BER]);
      OprStatus++;
      if (holdingRegs[HOST_EVENT_NUMBER] ==
tempEvent.NomorEvent){  //Valid event number
       OprStatus++;
       transferEventToHoldingRegs(tempEvent);
       OprStatus++;
      }
    }
   break;
   case CLEAR_EVENT:  //Hapus semua event
    OprStatus++;
    if (holdingRegs[HOST_EVENT_NUMBER] > 0 ){
     startAddr = ALAMAT_EEPROM_AWAL_EVENT - 1;
     lenghtAddr = JUMLAH_EVENT_MAX * sizeof(DataEvent);
//Event histories
     OprStatus++;
     ClearEEPROM(startAddr,lenghtAddr);
     holdingRegs[HOST_EVENT_NUMBER] = 0;
     OprStatus++;
    }
```

```
    break;
    default:
    break;
   }
  if (LastOprStatus < OprStatus)LastOprStatus = OprStatus;
//Ambil status tertinggi
//Operation has been completed with last status recorded
  if (HostCommand  > NO_OPERATION)  {
    if (LastOprStatus >= OprStatus){
      holdingRegs[HOST_OPR_STATUS]  = LastOprStatus;
      holdingRegs[LAST_OPR_MODE]  = HostCommand;
      holdingRegs[HOST_OPR_MODE]  = NO_OPERATION;
    }
  }
}

void transferEventToHoldingRegs(DataEvent tempEvent){

holdingRegs[HOST_EVENT_NUMBER]=tempEvent.NomorEven
t;
    holdingRegs[HOST_EVENT_TAHUN]=tempEvent.Tahun;
    holdingRegs[HOST_EVENT_BULAN]=tempEvent.Bulan;
    holdingRegs[HOST_EVENT_TGL]=tempEvent.Tanggal;
    holdingRegs[HOST_EVENT_JAM]=tempEvent.Jam;
    holdingRegs[HOST_EVENT_MENIT]=tempEvent.Menit;
    holdingRegs[HOST_EVENT_DETIK]=tempEvent.Detik;

holdingRegs[HOST_EVENT_ARUS]=tempEvent.Arus*WORD_
MAX/MAX_PV;

holdingRegs[HOST_EVENT_TEG]=tempEvent.Tegangan*WOR
D_MAX/MAX_TEG;
    holdingRegs[HOST_EVENT_STATUS]=tempEvent.Status;
    holdingRegs[HOST_EVENT_TYPE]=tempEvent.EventType;
}

void transferFromDataRelayToRelayShortCirt(){
  RelayShortCirt.setTimeChar(DataReleShortCircuit.TChar);
  RelayShortCirt.setActionChar(DataReleShortCircuit.Action);
```

```
  RelayShortCirt.setSetting(DataReleShortCircuit.SetPoint);
  RelayShortCirt.setTMS(DataReleShortCircuit.TMS);
  RelayShortCirt.setTimeDef(DataReleShortCircuit.TSet);
}

void transferFromDataRelayToRelayOverLoad(){
  RelayOverLoad.setTimeChar(DataReleOverLoad.TChar);
  RelayOverLoad.setActionChar(DataReleOverLoad.Action);
  RelayOverLoad.setSetting(DataReleOverLoad.SetPoint);
  RelayOverLoad.setTMS(DataReleOverLoad.TMS);
  RelayOverLoad.setTimeDef(DataReleOverLoad.TSet);
}

byte transferFromHostHoldingRegsToDataRelay(){
  byte stepFunction = 1;
//Data setting RelayShorCircuit
  DataReleShortCircuit.TChar  =
holdingRegs[HOST_SH_TCHAR];
  DataReleShortCircuit.Action  =
holdingRegs[HOST_SH_ACTION];
  DataReleShortCircuit.SetPoint  =
holdingRegs[HOST_SH_SP]*MAX_PV/WORD_MAX;
  DataReleShortCircuit.TMS  = holdingRegs[HOST_SH_TMS];
  DataReleShortCircuit.TSet  = holdingRegs[HOST_SH_TSET];

  transferFromDataRelayToRelayShortCirt();
  stepFunction++;//increase a step

//Data setting RelayOverLoad  22/05/2017
  DataReleOverLoad.TChar  = holdingRegs[HOST_OL_TCHAR];
  DataReleOverLoad.Action  = holdingRegs[HOST_OL_ACTION];
  DataReleOverLoad.SetPoint  =
holdingRegs[HOST_OL_SP]*MAX_PV/WORD_MAX;
  DataReleOverLoad.TMS  = holdingRegs[HOST_OL_TMS];
  DataReleOverLoad.TSet  = holdingRegs[HOST_OL_TSET];

  transferFromDataRelayToRelayOverLoad();
  stepFunction++; //increase a step
```

```
//penambahan Version dan waktu
  if (holdingRegs[HOST_SET_VERSION] >= 0xFF)
holdingRegs[HOST_SET_VERSION] = 1; //Kembali ke 1/roll
over
  DataReleShortCircuit.Version =
holdingRegs[HOST_SET_VERSION] ; //Di-update dari Host
  DataReleShortCircuit.Detik =SaatIni.second() ;
  DataReleShortCircuit.Menit = SaatIni.minute();
  DataReleShortCircuit.Jam = SaatIni.hour();
  DataReleShortCircuit.Tanggal = SaatIni.day();
  DataReleShortCircuit.Bulan = SaatIni.month();
  DataReleShortCircuit.Tahun = SaatIni.year();
  stepFunction++; //increase a step

//Data setting RelayOverLoad
  DataReleOverLoad.Version =
holdingRegs[HOST_SET_VERSION] ;//di-update dari Host
  DataReleOverLoad.Detik =SaatIni.second() ;
  DataReleOverLoad.Menit = SaatIni.minute();
  DataReleOverLoad.Jam = SaatIni.hour();
  DataReleOverLoad.Tanggal = SaatIni.day();
  DataReleOverLoad.Bulan = SaatIni.month();
  DataReleOverLoad.Tahun = SaatIni.year();

  stepFunction++; //increase a step

  EEPROM.put(ALAMAT_EEPROM_AWAL_DATA,
DataReleShortCircuit);
  stepFunction++; //increase a step
  EEPROM.put(ALAMAT_EEPROM_AWAL_DATA +
sizeof(DataSettingRelay), DataReleOverLoad);
  stepFunction++; //increase a step
  return stepFunction;
}

void updateDataRelayFromEEPROM(){
  boolean DataRelayValid = true;
  EEPROM.get(ALAMAT_EEPROM_AWAL_DATA,
DataReleShortCircuit);
```

```
  EEPROM.get(ALAMAT_EEPROM_AWAL_DATA +
sizeof(DataSettingRelay), DataReleOverLoad);

//check setting ShortCircuit Valid
  if ((DataReleShortCircuit.Version <=
0)|(DataReleShortCircuit.Version >= 0xFF)){
    DataRelayValid = false;
    DataReleShortCircuit.Version = 0;
    DataReleOverLoad.Version = 0;
  }
//check setting OverLoad Valid
  if ((DataReleOverLoad.Version <=
0)|(DataReleOverLoad.Version >= 0xFF)){
    DataRelayValid = false;
    DataReleShortCircuit.Version = 0;
    DataReleOverLoad.Version = 0;
  }
  if (DataRelayValid){
    transferFromDataRelayToRelayShortCirt();
    transferFromDataRelayToRelayOverLoad();

    holdingRegs[HOST_SET_VERSION] =
DataReleShortCircuit.Version;
    holdingRegs[HOST_SET_TAHUN] =
DataReleShortCircuit.Tahun;
    holdingRegs[HOST_SET_BULAN] =
DataReleShortCircuit.Bulan;
    holdingRegs[HOST_SET_TGL] =
DataReleShortCircuit.Tanggal;
    holdingRegs[HOST_SET_JAM] = DataReleShortCircuit.Jam;
    holdingRegs[HOST_SET_MENIT] =
DataReleShortCircuit.Menit;
    holdingRegs[HOST_SET_DETIK] =
DataReleShortCircuit.Detik;
  }
}
void ClearEEPROM(word startAddr, word lenghtAddr){
  word maxEEPROMaddr = startAddr + lenghtAddr;
  if (maxEEPROMaddr < EEPROM.length()){
```

```
    for (int i = 0 ; i < lenghtAddr; i++) {
      EEPROM.write(i+startAddr, 0);
    }
   }
}

/*MenuReleComb*/
/** \file
Relecomb menu sbb:
Menu Utama: -> Index 0
  Rele1 -> Trip-Reset Index -> Index 10
    Test Trip -> Index 14
    Reset -> Index 15
  DisPlay -> Index 20
  Status -> Index 30
  Info -> Index 40
*/
const char ReleCombMn0[] PROGMEM="Trip-Reset";
const char ReleCombMn1[] PROGMEM="Display";
const char ReleCombMn2[] PROGMEM="Status";
const char ReleCombMn3[] PROGMEM="Info";
const char* const ReleCombMnItems[] PROGMEM =
{ReleCombMn0, ReleCombMn1, ReleCombMn2,
ReleCombMn3};
phi_prompt_struct mainMenu; //This structure stores the main
menu.

//This program is the main menu. It handles inputs from the keys,
updates the menu or executes a certain menu function accordingly.
void mainMenuRele(){
   switch (menuId.mainMenu) //See which menu item is selected
and execute that correS_Pond function
    {
      case MAIN_MENU_IDX:
        menuReleComb();
      break;
      case TRIP_RESET_MENU_IDX:
        subMenuTripReset("<<-Trip-Reset->>"); //lihat file
subMenuRele
```

```
    break;
    case DISPLAY_MENU_IDX:
      subMenuDisPlay(); //Menampilkan tegangan dan arus
setting/actual
    break;
    case STATUS_MENU_IDX:
      subMenuStatus(); //Menampilkan tgl,bln,thn jam:mm:ss  dan
status relay
    break;
    case INFO_MENU_IDX:
      subMenuInfo();
    break;
    default:
    break;
  }
}

void setupMenuReleComb(){
  mainMenu.ptr.list=(char**)&ReleCombMnItems;  // Assign the
list to the pointer
  mainMenu.low.i=0;  // Default item highlighted on the list
  mainMenu.high.i=2;  // Last item of the list is size of the list - 1.
  mainMenu.width=LCD_COLUMNS-
((global_style&phi_prompt_arrow_dot)!=0)-
((global_style&phi_prompt_scroll_bar)!=0);  // Auto fit the size of
the list to the screen. Length in characters of the longest list item.
  mainMenu.step.c_arr[0]=LCD_ROWS-1;  // rows to auto fit entire
screen
  mainMenu.step.c_arr[1]=1;  // one col list
  mainMenu.step.c_arr[2]=0;  // y for additional feature such as an
index
  mainMenu.step.c_arr[3]=LCD_COLUMNS-4-
((global_style&phi_prompt_index_list)!=0);  // x for additional
feature such as an index
  mainMenu.col=0;  // DisPlay menu at column 0
  mainMenu.row=1;  // DisPlay menu at row 1
  mainMenu.option=global_style; // Option 0, disPlay classic list,
option 1, disPlay 2X2 list, option 2, disPlay list with index, option
3, disPlay list with index2.
```

```
}

void menuReleComb()
{
  int menu_pointer_1=0; // This stores the menu choice the user
made.
  lcd.clear();  // Refresh menu if a button has been pushed
  center_text("Main Menu");//Menu Title

  select_list(&mainMenu); // Use the select_list to ask the user to
select an item of the list, that is a menu item from your menu.
  menu_pointer_1=mainMenu.low.i;  // Get the selected item
number and store it in the menu pointer.
  switch (menu_pointer_1) // See which menu item is selected and
execute that correS_Pond function
  {
   case 0:
     menuId.mainMenu = TRIP_RESET_MENU_IDX;
     menuId.subMenu = 1;
   break;
   case 1:
     menuId.mainMenu = DISPLAY_MENU_IDX;
     menuId.subMenu = 1;
   break;
   case 2:
     menuId.mainMenu = STATUS_MENU_IDX;
     menuId.subMenu = 1;
   break;
   case 3:
     menuId.mainMenu = INFO_MENU_IDX;
     menuId.subMenu = 1;
   break;
   default:
   break;
  }
}

void subMenuDisPlay(){
  byte currentKey;
```

```
  String Sval;
  lcd.clear();  // Refresh menu if a button has been pushed

//Tampilkan pada baris ke 1
  Sval = String("Set:");
  Sval = String(Sval + RelayShortCirt.getSetting()); //EmonLib
Arus SC
  Sval = String(Sval + "/");
  Sval = String(Sval + RelayOverLoad.getSetting()); //EmonLib
Arus OL
  Sval = String(Sval + " A");
  lcd.setCursor(0,0);//posisikan kursor pada baris 1 kolom 1
  lcd.print(Sval);

//Tampilkan Nilai Tegangan pada baris ke 2
  Sval = String("V/I:");
  Sval = String(Sval + monitoringArusDanTegangan.Vrms);
//EmonLib Tegangan
  Sval = String(Sval + "/");
  Sval = String(Sval + RelayShortCirt.getValue());  //EmonLib
Arus
  lcd.setCursor(0,1); //Posisikan kursor pada baris 2 kolom 1
  lcd.print(Sval);

  currentKey = analogKeypad.getKey(); // Use phi_keypads object
to access the keypad
  switch (currentKey) // See which menu item is selected and
execute that correS_Pond function
  {
   case 'S':
     menuId.mainMenu = MAIN_MENU_IDX;
     menuId.subMenu = 1;
     return;
   break;
   default:
   break;
  }
}
```

```
void subMenuStatus(){
  // menu untuk menampilkan  tgl dan waktu pada baris pertama
  // status relay pada baris kedua
  byte currentKey;
  String Sval;
  lcd.clear();  // Refresh menu if a button has been pushed

//Tampilkan  pada baris ke 1
  Sval =  String(SaatIni.day());//ambil  tanggal
  Sval =  String(Sval + "/");
  Sval =  String(Sval + SaatIni.month());//ambil  bulan
  Sval =  String(Sval + "/");
  Sval =  String(Sval + (SaatIni.year()-2000));////ambil  tahun
  Sval =  String(Sval + " ");
  Sval =  String(Sval + SaatIni.hour());////ambil  jam
  Sval =  String(Sval + ":");
  Sval =  String(Sval + SaatIni.minute());////ambil  menit
  Sval =  String(Sval + ":");
  Sval =  String(Sval + SaatIni.second());////ambil detik

  lcd.setCursor(0,0);
  lcd.print(Sval);

//Tampilkan  status relay pada baris ke 2
  lcd.setCursor(0,1);
  lcd.print(statusRelay());
  currentKey = analogKeypad.getKey(); // Use phi_keypads object
to access the keypad
  switch (currentKey) // See which menu item is selected and
execute that correS_Pond function
  {
    case 'S':
      menuId.mainMenu  = MAIN_MENU_IDX;
      menuId.subMenu = 1;
      return;
    break;
    default:
    break;
  }
```

```
}

String statusRelay(){
  byte _state;
  String Sval;
  _state = RelayShortCirt.getState();
  if (_state < RelayOverLoad.getState())_state =
RelayOverLoad.getState(); //ambil nilai status tertinggi
  Sval = String("STS:");
  if (_state == STATUS_OK)return (Sval = String(Sval + "OK"));
  else if (_state >= STATUS_TRIP){
    Sval = String(Sval + "TRP->");
    _state = _state - STATUS_TRIP;
  }
  if (_state > STATUS_OK){//Trip  atau belum trip dengan
beberapa status
    //check status Relay
    switch (_state){
      case STATUS_OVER:
        return String(Sval + "OVR");
      break;
      case STATUS_UNDER:
        return String(Sval + "UDR");
      break;
      case EQL_ACTION:
        return String(Sval + "EQL");
      break;
      case STATUS_TEST_LOCAL:
        return String(Sval + "LOCAL");
      break;
      case STATUS_TEST_KEY:
        return String(Sval + "KEYPAD");
      break;
      case STATUS_TEST_REMOTE:
        return String(Sval + "REMOTE");
      break;
      default:
      break;
    }
```

```
  }
}

void subMenuInfo(){
char infoMsg[]="by: Abi and Sindhu ";
char buffer[15];
lcd.clear();
lcd.noBlink();
center_text("Protection Relay"); // display judul
for (byte i=0;i<strlen(infoMsg);i++)
  {
    scroll_text(infoMsg,buffer,14,i-14);
    lcd.setCursor(1,1);
    lcd.print(buffer);
    wait_on_escape(300);
  }
  menuId.mainMenu  = DISPLAY_MENU_IDX;
  menuId.subMenu = 1;
}

void subMenuTripReset(char* judulMenu)
{
  byte currentKey;
  lcd.clear();   // Refresh menu if a button has been pushed
  center_text(judulMenu);//Menu Title
  currentKey = analogKeypad.getKey(); // Use phi_keypads object
to access the keypad
  switch (currentKey) // See which menu item is selected and
execute that correS_Pond function
  {
    case 'U':
      //menuId.mainMenu  = TRIP_RESET_MENU_IDX;
      menuId.subMenu--;
      if (menuId.subMenu <  TRIP_MENU_IDX)  menuId.subMenu
=  BACK_MENU_IDX;   //roll over to Back_maneu
    break;
    case 'D':
      menuId.subMenu++;
```

```
      if (menuId.subMenu >  BACK_MENU_IDX)
menuId.subMenu =  TRIP_MENU_IDX;   //roll over to Trip_menu
    break;
    case 'S':
      if (menuId.subMenu ==  TRIP_MENU_IDX){
        menuId.mainMenu  = 0;
        RelayShortCirt.setState(STATUS_TEST_KEY);   //Test trip
dari keypad (SC)
        RelayOverLoad.setState(STATUS_TEST_KEY);   //Test trip
dari keypad (OL)
        return;//ini untuk test trip
      }
      if (menuId.subMenu ==  RESET_MENU_IDX){
        menuId.mainMenu  = 0;
        RelayShortCirt.setReset(true);  //Reset dari keypad (SC)
        RelayOverLoad.setReset(true); //Reset dari keypad (OL)
        return;//ini untuk Reset
      }
      if (menuId.subMenu ==  BACK_MENU_IDX){
        menuId.mainMenu  = 0;
        return;
      }
    break;
    default:
    break;
  }
  lcd.clear();   // Refresh menu if a button has been pushed
  center_text(judulMenu);//Menu Title
  DisplayRelayLCD();
}

void DisplayRelayLCD(){
  String Sval;
  byte _timeChar;
  if (menuId.subMenu == TRIP_MENU_IDX){
    Sval =  String("Test Trip");
  }
  if (menuId.subMenu == RESET_MENU_IDX){
    Sval =  String("Reset");
```

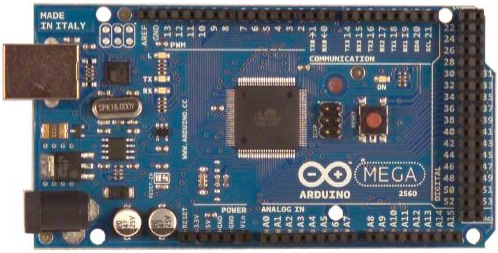
```
  }
  if (menuId.subMenu == BACK_MENU_IDX){
    Sval =  String("Back To Main Menu");
  }
  lcd.setCursor(0,1);
  lcd.print(Sval);
}
```

LAMPIRAN B

B.1 DATASHEET ARDUINO MEGA

ARDUINO

Arduino Mega 2560 Datasheet



Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode() , digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH

value, the LED is on, when the pin is LOW, it's off.

- **I2C: 20 (SDA) and 21 (SCL).** Support I2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I2C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).
A SoftwareSerial library allows for serial communication on any of the Mega2560's digital pins.
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. For SPI communication, use the SPI library.

Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference and tutorials.
The ATmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It

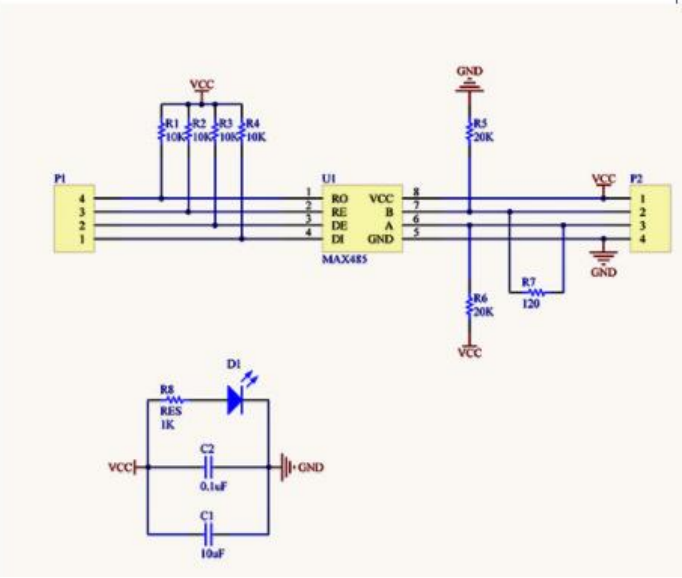
communicates using the original STK500 protocol (reference, C header files).
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

B.2 DATASHEET RS485

How-To Information Link HERE:

See Example RS485 Network Diagram below.

SCHEMATIC OF THIS MODULE:



Selection Table

PART NUMBER	HALF/FULL DUPLEX	DATA RATE (Mbps)	SLEW-RATE LIMITED	LOW-POWER SHUTDOWN	RECEIVER/ DRIVER ENABLE	QUIESCENT CURRENT (µA)	NUMBER OF RECEIVERS ON BUS	PIN COUNT
MAX481	Half	2.5	No	Yes	Yes	300	32	8
MAX483	Half	0.25	Yes	Yes	Yes	120	32	8
MAX485	Half	2.5	No	No	Yes	300	32	8
MAX487	Half	0.25	Yes	Yes	Yes	120	128	8
MAX488	Full	0.25	Yes	No	No	120	32	8
MAX489	Full	0.25	Yes	No	Yes	120	32	14
MAX490	Full	2.5	No	No	No	300	32	8
MAX491	Full	2.5	No	No	Yes	300	32	14
MAX1487	Half	2.5	No	No	Yes	230	128	8

For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim Integrated's website at www.maximintegrated.com.

19-0122; Rev 10; 9/14

B.3 DATASHEET RTC DS1307

DALLAS SEMICONDUCTOR MAXIM

DS1307
64 x 8, Serial, I²C Real-Time Clock

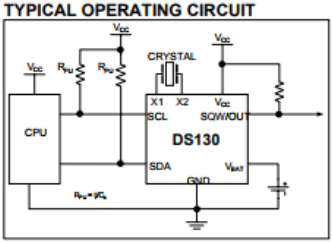
GENERAL DESCRIPTION

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I²C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.

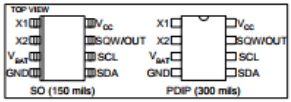
FEATURES

- Real-Time Clock (RTC) Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the week, and Year with Leap-Year Compensation Valid Up to 2100
- 56-Byte, Battery-Backed, General-Purpose RAM with Unlimited Writes
- I²C Serial Interface
- Programmable Square-Wave Output Signal
- Automatic Power-Fail Detect and Switch Circuitry
- Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running
- Optional Industrial Temperature Range: -40°C to +85°C
- Available in 8-Pin Plastic DIP or SO
- Underwriters Laboratories (UL) Recognized

TYPICAL OPERATING CIRCUIT



PIN CONFIGURATIONS



ORDERING INFORMATION

PART	TEMP RANGE	VOLTAGE (V)	PIN-PACKAGE	TOP MARK*
DS1307+	0°C to +70°C	5.0	8 PDIP (300 mils)	DS1307
DS1307N+	-40°C to +85°C	5.0	8 PDIP (300 mils)	DS1307N
DS1307Z+	0°C to +70°C	5.0	8 SO (150 mils)	DS1307
DS1307ZN+	-40°C to +85°C	5.0	8 SO (150 mils)	DS1307N
DS1307Z+T&R	0°C to +70°C	5.0	8 SO (150 mils) Tape and Reel	DS1307
DS1307ZN+T&R	-40°C to +85°C	5.0	8 SO (150 mils) Tape and Reel	DS1307N

+Denotes a lead-free/RoHS-compliant package.
**A "+" anywhere on the top mark indicates a lead-free package. An "N" anywhere on the top mark indicates an industrial temperature range device.*

ABSOLUTE MAXIMUM RATINGS

Voltage Range on Any Pin Relative to Ground .....-0.5V to +7.0V
Operating Temperature Range (Noncondensing)
Commercial.....0°C to +70°C
Industrial .....-40°C to +85°C
Storage Temperature Range ..... -55°C to +125°C
Soldering Temperature (DIP, leads).....+260°C for 10 seconds
Soldering Temperature (surface mount).....Refer to the JPC/JEDEC J-STD-020 Specification.

Stresses beyond those ilisted under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to the absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED DC OPERATING CONDITIONS

(T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Supply Voltage	V_{CC}		4.5	5.0	5.5	V
Logic 1 Input	V_{IH}		2.2		V_{CC} + 0.3	V
Logic 0 Input	V_{IL}		-0.3		+0.8	V
V_{BAT} Battery Voltage	V_{BAT}		2.0	3	3.5	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = **4.5V to 5.5V**; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Input Leakage (SCL)	I_{LI}		-1		1	µA
I/O Leakage (SDA, SQW/OUT)	I_{LO}		-1		1	µA
Logic 0 Output (I_{OL} = 5mA)	V_{OL}				0.4	V
Active Supply Current (f_{SCL} = 100kHz)	I_{CCA}				1.5	mA
Standby Current	I_{CCS}	(Note 3)			200	µA
V_{BAT} Leakage Current	I_{BATLKG}			5	50	nA
Power-Fail Voltage (V_{BAT} = 3.0V)	V_{PF}		1.216 x V_{BAT}	1.25 x V_{BAT}	1.284 x V_{BAT}	V

DC ELECTRICAL CHARACTERISTICS

(V_{CC} = **0V**, V_{BAT} = **3.0V**; T_A = 0°C to +70°C, T_A = -40°C to +85°C.) (Notes 1, 2)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
V_{BAT} Current (OSC ON); SQW/OUT OFF	I_{BAT1}			300	500	nA
V_{BAT} Current (OSC ON); SQW/OUT ON (32kHz)	I_{BAT2}			480	800	nA
V_{BAT} Data-Retention Current (Oscillator Off)	I_{BATDR}			10	100	nA

WARNING: Negative undershoots below -0.3V while the part is in battery-backed mode may cause loss of data.

AC ELECTRICAL CHARACTERISTICS

(V_{CC} = 4.5V to 5.5V; T_A = 0°C to +70°C; T_A = -40°C to +85°C.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
SCL Clock Frequency	f_{SCL}		0		100	kHz
Bus Free Time Between a STOP and START Condition	t_{BUF}		4.7			µs
Hold Time (Repeated) START Condition	$t_{HD:STA}$	(Note 4)	4.0			µs
LOW Period of SCL Clock	t_{LOW}		4.7			µs
HIGH Period of SCL Clock	t_{HIGH}		4.0			µs
Setup Time for a Repeated START Condition	$t_{SU:STA}$		4.7			µs
Data Hold Time	$t_{HD:DAT}$		0			µs
Data Setup Time	$t_{SU:DAT}$	(Notes 5, 6)	250			ns
Rise Time of Both SDA and SCL Signals	t_R				1000	ns
Fall Time of Both SDA and SCL Signals	t_F				300	ns
Setup Time for STOP Condition	$t_{SU:STO}$		4.7			µs

CAPACITANCE

(T_A = +25°C)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Pin Capacitance (SDA, SCL)	$C_{I/O}$				10	pF
Capacitance Load for Each Bus Line	C_B	(Note 7)			400	pF

Note 1: All voltages are referenced to ground.

Note 2: Limits at -40°C are guaranteed by design and are not production tested.

Note 3: I_{CCS} specified with V_{CC} = 5.0V and SDA, SCL = 5.0V.

Note 4: After this period, the first clock pulse is generated.

Note 5: A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the $V_{IH(MIN)}$ of the SCL signal) to bridge the undefined region of the falling edge of SCL.

Note 6: The maximum $t_{HD:DAT}$ only has to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.

Note 7: C_B—total capacitance of one bus line in pF.

TIMING DIAGRAM

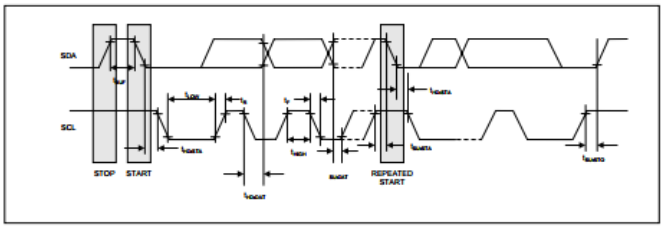
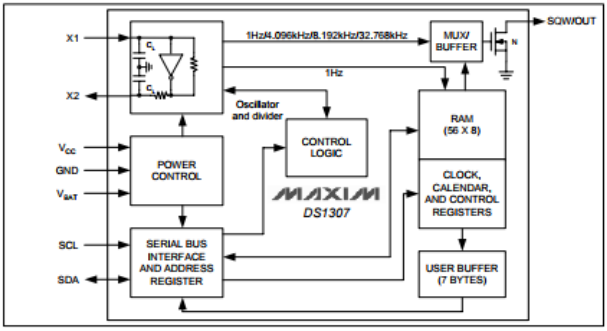


Figure 1. Block Diagram



B.4 DATASHEET YHDC SCT 13-010

Product Specification Date:2015-8-7

Product Name	Current transformer	Model	SCT013-010

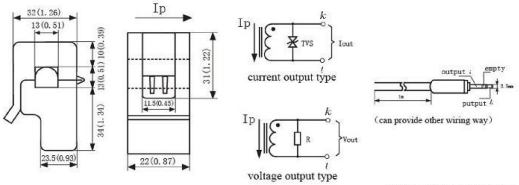
Characteristics:Openning size 13mm*13mm,1m leading wire, standard Φ3.5 three-core plug output. Have two kinds of output type: Current output type and voltage output type.
Purpose: Used for current measurement, monitor and protection for AC motor,lighting equipment, air compressor etc

Technical Data

I_{PN}	Rated input	0-10A
I_{PM}	Max. detection input	
I_{OUT}	Rated output	0-1V
X	Accuracy	±1%
ε_L	Linearity	≤0.2%
N	Turns ratio	1:1800
Φ	Phase shift	
R_L	Max.Sampling resistance	
V_{PN}	Work voltage	660V
f	Work frequency	50-1KHz
T_A	Operating temperature	-25..+70℃
T_S	Storage temperature	-40..+85℃
Vd	Dielectric strength, 50 Hz, 1 min	3KV

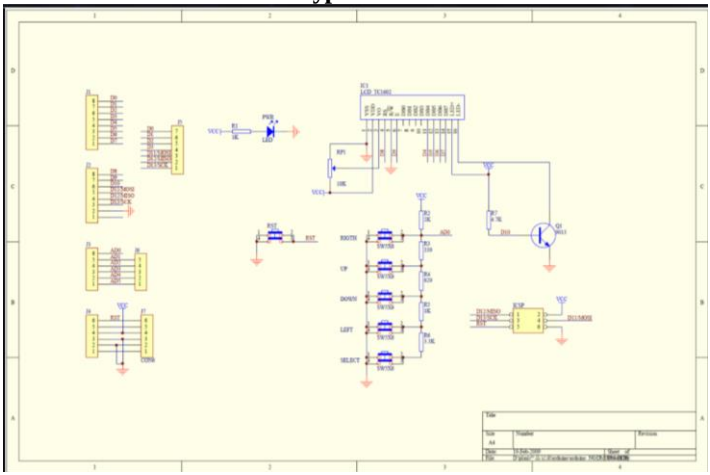
Fire resistance	UL94-V0
Material of core	Ferrite
Mounting type	Suspension
Weight	55g

Dimension (mm(in). 1 mm= 0.0394 inch)

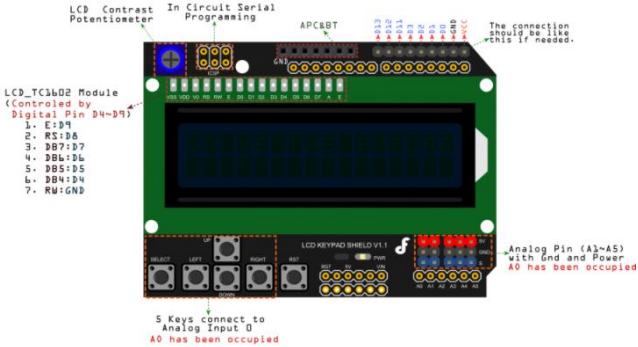


Front view Side view Schematic diagram Standard three-core pligs schematic diagram

B.5 DATASHEET LCD Keypad Shield



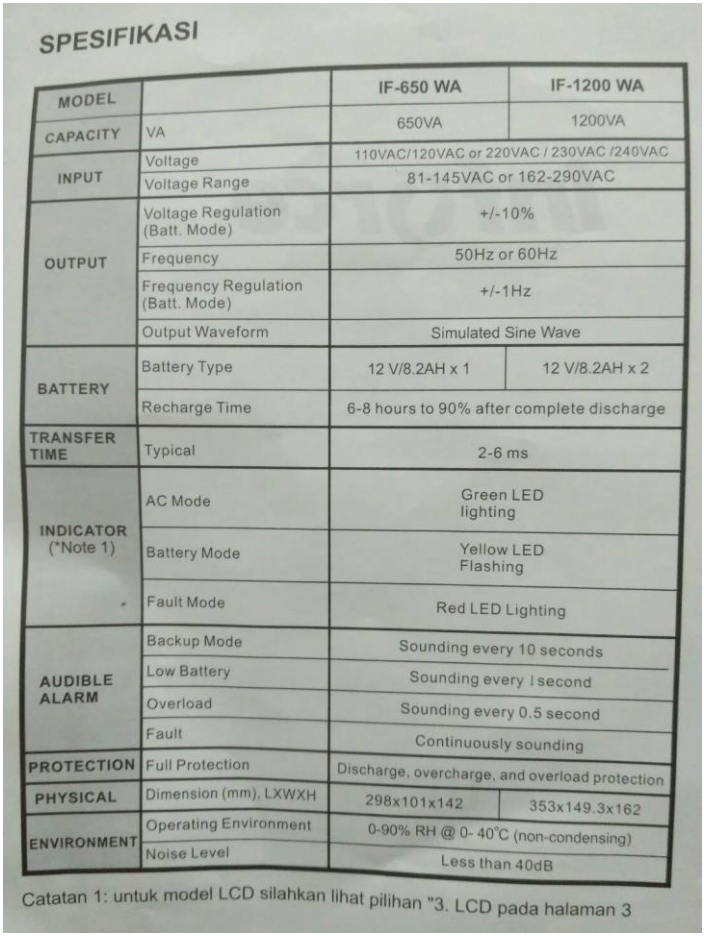
Pinout



Instruction for D4 To D10 and Analog Pin 0

Pin	Function	Instruction
Digital 4(D4)		
Digital 5(D5)	D4~D7 are used as DB4~DB7	Four high order bidirectional tristate data bus pins. Used for data transfer and receive between the MPU and the LCD.
Digital 6(D6)		
Digital 7(D7)		
Digital 8(D8)	RS	Choose Data or Signal Display
Digital 9(D9)	Enable	Starts data read/write
Digital 10(D10)	LCD Backlight Control	
Analog 0(A0)	Button select	Select, up, right, down and left

B.6 Datasheet UPS



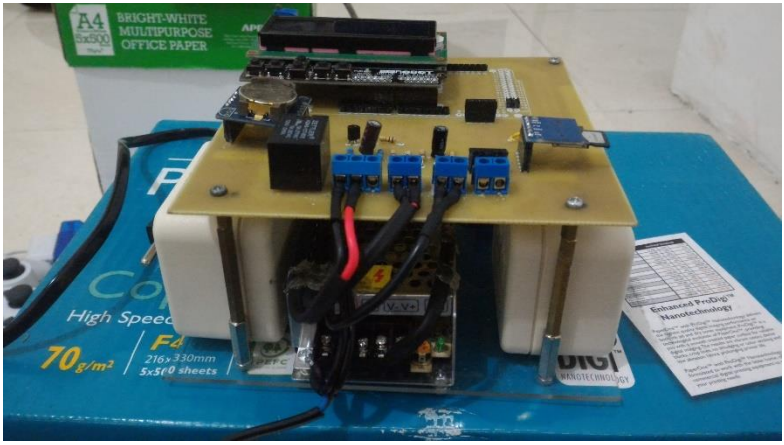
SPESIFIKASI

MODEL		IF-650 WA	IF-1200 WA
CAPACITY	VA	650VA	1200VA
INPUT	Voltage	110VAC/120VAC or 220VAC / 230VAC /240VAC	
	Voltage Range	81-145VAC or 162-290VAC	
OUTPUT	Voltage Regulation (Batt. Mode)	+/-10%	
	Frequency	50Hz or 60Hz	
	Frequency Regulation (Batt. Mode)	+/-1Hz	
	Output Waveform	Simulated Sine Wave	
BATTERY	Battery Type	12 V/8.2AH x 1	12 V/8.2AH x 2
	Recharge Time	6-8 hours to 90% after complete discharge	
TRANSFER TIME	Typical	2-6 ms	
INDICATOR (*Note 1)	AC Mode	Green LED lighting	
	Battery Mode	Yellow LED Flashing	
	Fault Mode	Red LED Lighting	
AUDIBLE ALARM	Backup Mode	Sounding every 10 seconds	
	Low Battery	Sounding every 1 second	
	Overload	Sounding every 0.5 second	
	Fault	Continuously sounding	
PROTECTION	Full Protection	Discharge, overcharge, and overload protection	
PHYSICAL	Dimension (mm), LXWXH	298x101x142	353x149.3x162
ENVIRONMENT	Operating Environment	0-90% RH @ 0- 40°C (non-condensing)	
	Noise Level	Less than 40dB	

Catatan 1: untuk model LCD silahkan lihat pilihan "3. LCD pada halaman 3

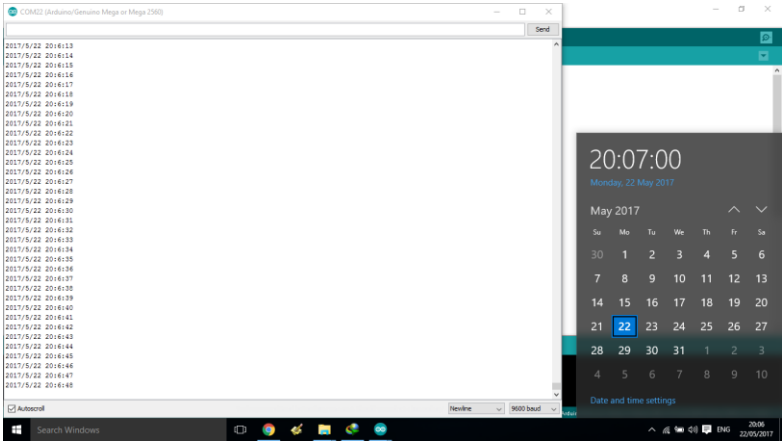
LAMPIRAN C

C.1 TAMPILAN RELAI

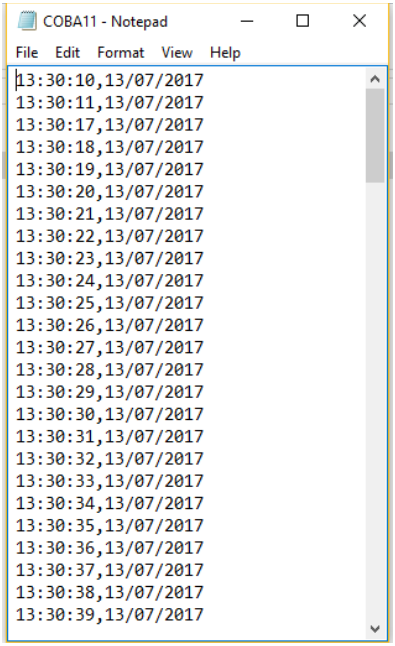


C.2 PENGUJIAN SENSOR ARUS DAN TEGANGAN

C.3 PENGUJIAN RTC

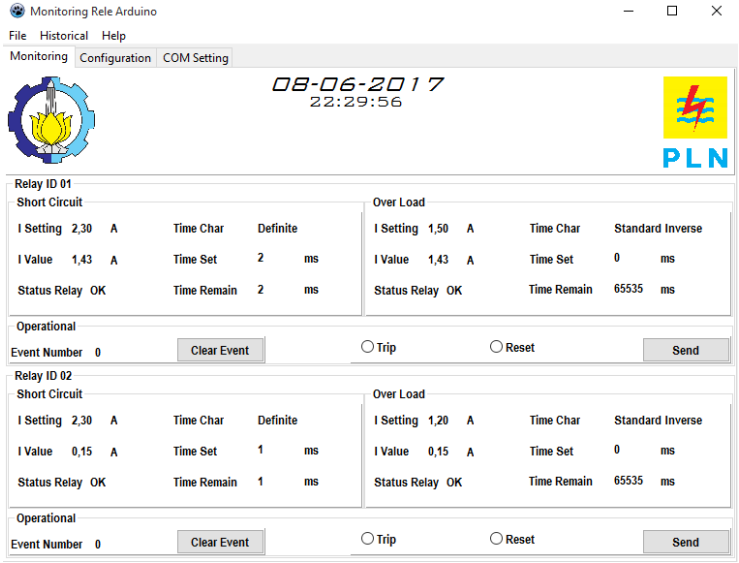


C.4 PENGUJIAN MEMORI SD CARD



C.5 PENGUJIAN RELAI

C.6 PENGUJIAN INTERFACE LAZARUS



DAFTAR RIWAYAT HIDUP

Nama	:	Abi Nubli
TTL	:	Serang, 04 September 1996
Jenis Kelamin	:	Laki - Laki
Agama	:	Islam
Alamat	:	Lidah RT/RW 02/08 Desa Gambiran Kec. Gambiran Kab. Banyuwangi
Telp/HP	:	087755873458
E-mail	:	abinbl01@gmail.com

RIWAYAT PENDIDIKAN

1. 2002 – 2008 : SD Muhammadiyah GKB Gresik
2. 2008 – 2011 : SMP Muhammadiyah 12 GKB Gresik
3. 2011 – 2014 : SMA Negeri 1 Genteng Banyuwangi
4. 2014 – 2017 : D3 Teknik Elektro Otomasi, Program Studi Teknik Listrik – Fakultas Vokasi Institut Teknologi Sepuluh Nopember (ITS)

PENGALAMAN KERJA

1. Kerja Praktek di PT PLN (Persero) Area Surabaya Selatan Jawa Timur

-----Halaman ini sengaja dikosongkan-----